Wearable Electronic Device Design for Preventive Health Care-Related Purposes

Todor Aleksandrov Ginchev

School of Electrical Engineering

Thesis submitted for examination for the degree of Master of Science in Technology.

Espoo October 27, 2015

Thesis supervisor:

Prof. Jörg Ott

Thesis advisors:

Prof. Kari Halonen

Dr. José Costa-Requena

Aalto University
School of Electrical
Engineering

AALTO UNIVERSITY
SCHOOL OF ELECTRICAL ENGINEERING

ABSTRACT OF THE
MASTER'S THESIS

Author: Todor Aleksandrov Ginchev

Title: Wearable Electronic Device Design for Preventive Health Care-Related Purposes

Date: October 27, 2015 Language: English Number of pages: 6+63

Department of Micro and Nanoscience

Professorship: Integrated Circuit Design

Supervisor: Prof. Jörg Ott

Advisors: Prof. Kari Halonen, Dr. José Costa-Requena

Medical diagnosis and healthcare recommendations are often challenging and require the correlation between different health-related inputs from the subject. On the one hand, some of the biometric data inputs are only possible to be determined by 24/7 monitoring, such as physical activity tracking, sleep quality or food intake nutritional information and quantity estimation. On the other hand, it is a big data problem as the correlation between all the inputs is rather challenging and time-consuming to be done by a human being.

This work presents the design of wearable wristband electronic device, capable of continuous monitoring of several biometric inputs. The device connects to the Internet via smartphone and sends the data to a server in a secure way, using proper authentication and personal data protection. The biometric data will be available for physicians and at the same time a machine learning algorithm will elaborate and send healthcare related recommendations to the user.

The wristband device is capable of tracking the physical activity, sleep quality and food intake of the subject. This is be done by several built-in sensors such as accelerometer, gyroscope, heart-rate sensor and digital camera. The accelerometer is be used to track the physical activity and the gyroscope detects wrist motion in order to recognize when the subject is eating and count the taken bites. Furthermore, the heart-rate sensor detects stress situations and the built-in camera takes a picture of the food. Thus, the picture is send to the smartphone via Bluetooth and then a computer vision algorithm is used to recognize the food.

The designed smart wristband circuit is first tested on a protoboard and then on an Arduino nano board. After that, the circuit is fabricated on a RF4 substrate PCB in order to test the final design. After verification, the circuit is finally fabricated on a flexible circuit, which concludes with the wristband prototype design, fabrication and verification.

Keywords: healthcare, smart wristband, wearable, activity monitoring, computer vision, digital image processing

Preface

The research leading to this thesis has been mainly carried out in the Department of Communications and Networking at the Aalto University School of Electrical Engineering.

First and foremost, I would like to express my sincere respect to my supervisor Prof. Jörg Ott for being critic with my work and at the same time allowing me to learn from my mistakes. Your feedback was very appreciated and useful.

I am thankful to my thesis advisor Prof. Kari Halonen for his technical support and for helping to solve all my issues in a very kind way.

I am really grateful to Dr. José Costa-Requena, who has been more a friend for me than an instructor. You helped me in the most important part of my life. Without you, I would not be able to come to Finland, develop myself technically and personally, learn so many interesting and useful things and share ideas with so many people. Thank you for letting me take part of a research team where one can easily learn from professionals from different sectors.

I would like to thank Jesús LLorente Santos, for helping me in my work, for his ability to motivate people by never telling them that they have done a good job, but especially for all the free chocolate. For me, you have been the workmate that I always wanted to have.

I would also like to thank my friends and workmates, especially Vicent, Mäel, Saray, Bruno, Antti and Nicolas Cage. For their help, good ideas but mostly for all the funny moments. It has been a pleasure and honour to work with you during the last year.

Special acknowledgements go to my parents, for believing in me and helping me with everything they have.

Finally, I would like to dedicate this work to my love Tedi, for supporting me in my decisions and being so lovely and always there for me in the hardest moments.

Otaniemi, October 27, 2015,

Todor Aleksandrov Ginchev

Contents

Abstract ii

Preface iii

Contents iv

Abbreviations vi

1 Introduction 1
- 1.1 The PRECIOUS Project concept . . . 2
- 1.2 Objective and scope . . . 3
- 1.3 Structure . . . 3

2 Background 4
- 2.1 Related work . . . 4
- 2.2 The Virtual Individual Model . . . 5
- 2.3 Physical activity monitoring . . . 5
  - 2.3.1 Step counter algorithm . . . 5
  - 2.3.2 Food intake habits tracking and bite counter algorithm . . . 6
- 2.4 Omnivision OV7670 digital camera data protocol . . . 7
- 2.5 Digital image processing applied to food detection and recognition . . . 9
- 2.6 Optical heart rate sensor design . . . 10
- 2.7 Flexible electronic circuit fabrication methods related to wearable devices . . . 11
- 2.8 Summary . . . 12

3 Smartphone application development 13
- 3.1 Physical activity tracker . . . 13
- 3.2 Weight estimation . . . 14
- 3.3 Food intake monitoring . . . 14
- 3.4 Food image detection and recognition . . . 15
  - 3.4.1 Applying Automatic colour balance . . . 15
  - 3.4.2 Colour filtering . . . 16
  - 3.4.3 Object segmentation . . . 22
  - 3.4.4 Contour detection and pattern matching . . . 22
- 3.5 Summary . . . 22

4 Wearable design, fabrication, programming and verification 25
- 4.1 Preliminary design on a proto-board . . . 25
  - 4.1.1 Integrated modules and other components . . . 25
  - 4.1.2 HC-05 Bluetooth V2.0 SPP module test . . . 26
  - 4.1.3 MPU-6050 3-axis gyroscope, 3-axis accelerometer and Digital Motion Processor test . . . 27

4.1.4 Omnivision OV7670 CMOS VGA camera module test 27
4.1.5 Heart-rate pulse sensor module test 29
4.2 Final prototype design on PCB board 29
4.2.1 Atmega328P connection and configuration 30
4.2.2 MPU-6050 MEMS module connection and configuration . . . 31
4.2.3 HC-06 Bluetooth module connection and configuration 31
4.2.4 Power supply management 32
4.3 Final schematics and PCB layout 33
4.4 Fabricating the wearable 33
4.4.1 PCB fabrication by the method of toner transfer 33
4.4.2 PCB fabrication by the method of UV-exposure 36
4.5 In-chip programming 39
4.6 Verifying the design and operation of the device 39
4.7 Summary 41

5 Conclusions **42**

References **44**

A Arduino code for OV7670 camera module frame reception **47**

B Python code that receives the image from the Arduino **50**

C Matlab code for processing the input image raw data **56**

D Wristband prototype source code **60**

Abbreviations

A/D	Analog to Digital
AWB	Automatic White Balance
BW	Bandwidth
DSP	Digital Signal Processor
FPC	Flexible Printed Circuit
HSV	Hue-Saturation-Value digital image colour space
LDO	Low-Dropout voltage regulator
LSB	Least Significant Bit
MCU	Microcontroller Unit
MEMS	Microelectromechanical Systems
PA	Physical Activity
PRECIOUS	PREventive Care Infrastructure based On Ubiquitous Sensing
PCB	Printed Circuit Board
RGB	Red-Green-Blue digital image colour model
VIM	Virtual Individual Model

1 Introduction

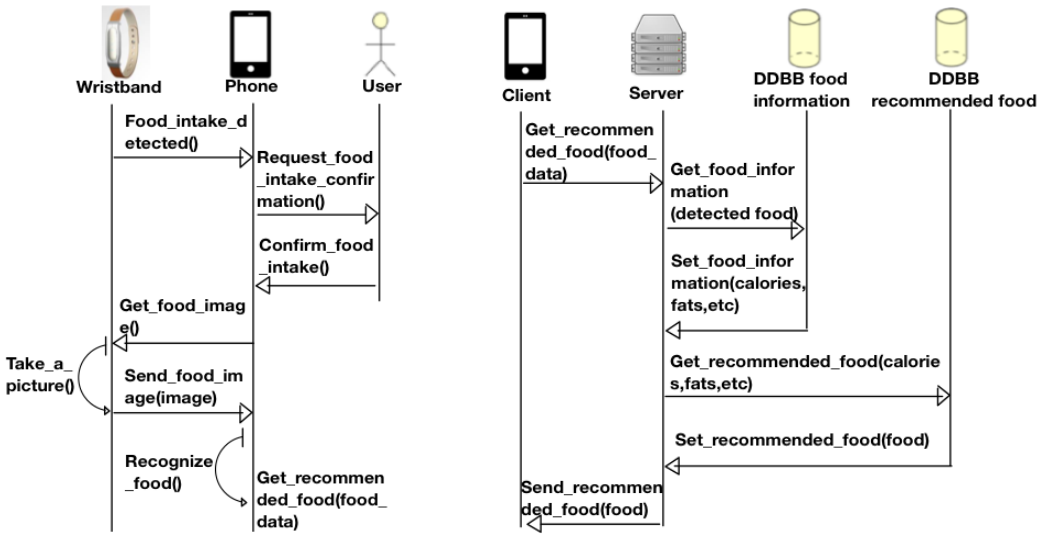
Medical diagnosis is often challenging and physicians are required to perform a very good linking of different signs, symptoms and other parameters involving the living being in order to compare them with already known diseases. It is a big data problem for a human being. However, for a machine it is just about the looking for the correlation between complex parameters and the further comparison against fuzzy data sets, which can be done by machine learning. Moreover, the continuous monitoring of the daily activity of a living being and its biometrics offers data that can be used to create a Virtual Individual Model (VIM), containing useful information that can lead to more accurate and faster diagnoses by applying machine learning.

The VIM parameters can be stored in a server and be easily obtained by a medical personal, always with a proper Internet authentication in order to achieve personal data privacy. Once on the server, this data can also be treated by different kind of machine learning software to achieve automatic everyday health-related recommendations and warnings.

For example, the tracking of eating habits is very important for the treatment of several maladies linked to obesity, such as diabetes. Wearable devices have been proved to be good for physical activity monitoring because of the low-cost and low-power built-in sensors. For research studies, this increases the number of subjects and the length of time that people can be monitored at a reduced cost. For the individual, it helps to alleviate the burden of self-monitoring, potentially increasing compliance and accuracy [1] [2]. Such devices can easily interconnect with a smartphone with Internet connection and send data to a centralized server. For example, in Figure 1 are illustrated the use cases of how a smart wristband with an integrated camera is able to send a photo to a server where the food taken with a built-in digital camera is recognized and nutritional information and food intake advices are send back to the user.

Moreover, wristband wearable devices track the physical activity of the user by using accelerometer MEMS sensor and they can also be used for fall detection [3]. This type of sensor can be used as a step counter or it can detect different kind of physical activities such as walking, running, riding a bicycle, etc, but others are more difficult to detect and therefore another kind of sensor is needed. For example, gyroscope MEMS sensor is suitable for wrists movement tracking and can be useful to track eating habits. Nevertheless, it has the disadvantage of using approximately ten times more power than accelerometer sensors. On the other hand, cameras are more complex optical sensors that offer the possibility to detect and recognize objects, such as food and beverages. Thus, by using these sensors in parallel, it is theoretically possible to track physical activity and food-intake habits, detect the food and extract the nutrition information by consulting a database.

Furthermore, heart-rate has been recognized as an indicator of health, disease, excitement and stress since the earliest days of medicine. Optical heart-rate sensors are suitable for continuous monitoring as they are non-invasive and very conformable to wear [4]. Although their main issue is that the accuracy is not as good as other products available on market, they are still suitable for 24/7 tracking.



(a) Bottom view. (b) Top view.

Figure 1: Figure 1a shows how the communication between the smart wristband and the user is done when implementing the food detection and recognition system. Figure 1b illustrates how the client connects to the databases that contain the nutritional information

It is therefore theoretically possible to monitor daily physical activity, food-intake and other human-being habits with a wearable device, such as wristband, in order to get important user data and create a VIM. The VIM is the heart of the PRECIOUS system which is briefly described in Section 1.1.

The main goals of this work and how do they fit in the PRECIOUS system are presented in Section 1.2. Finally, the structure of this document is detailed in Section 1.3.

1.1 The PRECIOUS Project concept

The PREventive Care Infrastructure based On Ubiquitous Sensing (PRECIOUS) project aims to improve the motivation of following healthier lifestyles by creating a personalised system, adapted to the user's goals and preferences.

On the one hand, several devices with sensors will monitor different biometrics of the user and send the information to the cloud. This includes environmental sensors which will track continuously the room temperature, humidity, light and noise or wearable devices capable of physical activity, stress level, location and food intake tracking.

By sending all the sensor data to the server, a Virtual Individual Model (VIM) is created. With the available information, machine learning algorithms are capable to provide a health care system that will improve the health of the user. This will also deliver cost savings in the public health sector [5].

1.2 Objective and scope

The main goal of this research is to study the viability of using a smart wristband device to track the physical activity, food-intake habits and other biometrics of a human being and how the device is going to interconnect with a smartphone and have access to the Internet, proving the concept of Internet of Things. This will be done by designing and implementing a server and a mobile software application, fabricating a wearable device prototype and defining important parameters related to the VIM. This will help to reveal the practical use-cases of such device and to set the prerequisites and preliminary design of the final device. The prototype must be therefore able to interact with a smartphone and send the collected data to a centralized server where a machine learning algorithm will look for health-related advices.

One the one hand, the smartphone application is a physical activity and food-intake monitor accessible for free to everyone that has access to an Android device. It will also try to detect health-related issues and improve the well-being of the user by applying motivation by gamification. It will connect to the Internet to send the collected data and receive recommended actions that the user should take for improving his or her health state.

On the other hand, the smart wristband is a device that will offer more accurate 24/7 monitoring, including sleep quality and heart-rate measurements. It will also have sensors for activity tracking and a camera for food recognition. It will interface a smartphone via Bluetooth and will send the food photos for their digital image processing which will be done either on the smartphone or on the PRECIOUS server.

The final objective is to achieve useful data for parametrizing a VIM in order to better detect and prevent healthcare related issues by using a portable smart device that will have access to the Internet.

To achieved the goals, first a health-related Android application will be developed. After that, the wearable prototype development will start by designing the circuit and implementing it on a prototype board. Once operational, the circuit will be fabricated on a PCB board. After the successful verification of the wristband, the goals of the research will be achieved.

1.3 Structure

This document is divided in 5 chapters. The second one is a state of the art analysis of the main principles of signal processing algorithms for physical activity tracking and food recognition. It also explains some basic circuit design and fabrication techniques. The third chapter describes the smartphone application development and the most important algorithms. The next chapter is a complete description of how the wearable was designed, starting from the proto-board circuit and then testing two different PCB fabrication techniques. Also, it demonstrates how the programming of the wearable was performed and how the operation verification was done. Finally, the fifth chapter is a summary of what have been done, the results and future work.

2 Background

Over the past decades, healthcare has evolved rapidly as an advanced technology with huge labour and monetary investments [6]. Nowadays, the usage e-Healthcare systems is increasing every day because more and more people find them useful. These systems can establish a profile for each user, storing important information regarding the user health-related habits and thus creating a Virtual Individual Model, which can be accessed by medical staff or the user himself.

This chapter starts by describing the VIM and what parameters form it. After that it is explained how some of the parameters can be obtained by a wearable device, emphasizing the physical activity and food intake tracking algorithms.

Finally, from the engineering point of view, the operation of a digital image camera is explained, followed by the electronic circuit of an optical heart-rate sensor and the discussion about the advantages of using FPC instead of PCB.

2.1 Related work

Biometric data tracking systems for physical activity and nutritional intake monitoring have been designed in different ways by using a vast variety of techniques. Most commonly, they are implemented on a smartphone or on a wearable device. Moreover, after sufficient data is obtained, there should be some feedback so that the subject can acknowledge his or her achievements. For instance, Liu uses a smartphone inertial sensors to detect different physical activities and then uses the data to for future recommendations about activity plans in order to motivate the user to keep fit [7]. On the other hand, in [8] Doron at proposes a design of a wearable device, capable of tracking the PA and estimate the energy expenditure by using a built-in accelerometer. The novelty of the wearable is that it uses an algorithm to recognize activities such as lying down, slouching, sitting down or cycling apart from the most typical ones that are walking and running. On the other hand, despite the fact that the typical sensors used for PA tracking are the accelerometer and sometimes the gyroscope, Shaopeng [9] utilizes a piezoelectric sensors to complement the wrist device that has the built-in accelerometer.

Moreover, the food intake monitoring using a wearable device is not so common nowadays, because it is still in the early research state. On the one hand, Shroff approach is to use smartphones camera and apply context-aware food recognition using artificial neural network [10]. On the other hand, in [11] Bi proposes the use acoustic sensing for nutritional intake tracking by designing a wearable with microphone which has to be worn on the neck. Furthermore, Jindong [12] combines these two techniques and builds a wearable with integrated camera and microphone, worn on the ear. In that way, it detects that the subject is eating by the chewing sound and starts taking photos of the food.

2.2 The Virtual Individual Model

The VIM is defined by health-related parameters that can be either measured by an electronic device or set manually. In the Android application, the input parameters involve the daily physical activity of the subject, geographical location, the nutritional information of the food intake and some personal data regarding the age, gender, height, weight and drinking or smoking habits. The wristband device includes more VIM input parameters like heart rate, sleeping quality and others. From the VIM inputs, other parameters can be determined indirectly, like stress level, mood, engagement, etc. as illustrated in Figure 2.

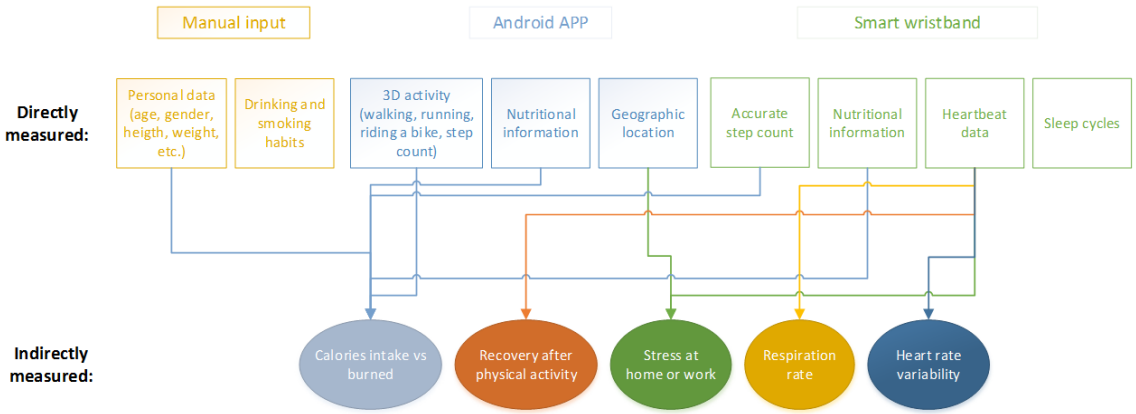


Figure 2: Some Virtual Individual Model parameters can be measured directly by the smart devices and others can be estimated. The diagram shows examples of those parameters and how are they measured.

2.3 Physical activity monitoring

The smart wristband physical activity tracker has two main purposes. On the one hand, step counting is done by processing the information from the accelerometer sensor and on the other hand, the food intake habits are detected with the help of the gyroscope sensor. This last is very useful as it helps to determine when the subject is eating and approximately how much bites did he or she is taking.

2.3.1 Step counter algorithm

The input data from the accelerometer is continuously processed. While walking, much differentiated gradients in the measured data occur and therefore the most straightforward way to detect steps is by looking for gradients and peaks. For instance, Orlando Hoilett at [13] proposes to look only in one of the axis, which direction is perpendicular to the surface the user is walking on. He measured the data from this axis, which turned to be the Y-axis on his case, and as a result he obtained the graphic shown in Figure 3.

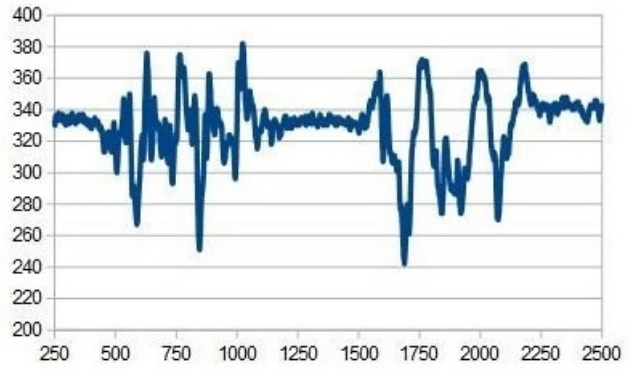


Figure 3: Y-axis accelerometer data measured by Orlando Hoilett at [13]. The first 500ms correspond to being still; from 500ms to 1000ms correspond to walking without swinging the arms; from 1000ms to 1500ms is still again; finally, the data from 1500ms to 2250ms corresponds to walking while swinging the arms.

It is therefore possible to implement a pedometer by simply tracking one of the axis and looking for gradients and peaks. It is also very simple and requires low processing power which is very important for a wearable device. Unfortunately, processing the data from only one of the axis leads to false positives, meaning that steps are detected when the subject is not walking, and also around 67% of the data from the accelerometer is wasted. Important improvement in this step counting method can be done by implementing a low-pass filter to reduce the noise [14]. Furthermore, the one-axis limitation can be removed by applying the Pythagorean Theorem over the 3 axis which will also make the device no orientation dependent.

More accurate method for implementing a pedometer is to compare the sensor data with an already known data. This will eliminate the false positives but it requires a memory for the training data and more processing, making it not suitable for wearable device where both memory and power are limited.

2.3.2 Food intake habits tracking and bite counter algorithm

The rotational velocity is a key measurement for tracking a human-being wrist in order to detect eating habits. A gyroscope is able to measure it along the 3 axis as illustrated in Figure 4, where the hand's rotation dimensions are known as roll, pitch and yaw. Furthermore, the roll axis is sufficient to track the wrist and look for eating activity patterns as shown in Figure 5.

The proposed algorithm for bite counting in [1] is not very different from the step counter algorithm, but it is less susceptible to false despite of processing the data from only one of the gyroscope rotational velocity axis. Using a state machine, every roll data sample is compared to the previous one and, depending on the current state, action is taken. This requires low processing power and memory and it is therefore suitable for a smart wristband device.

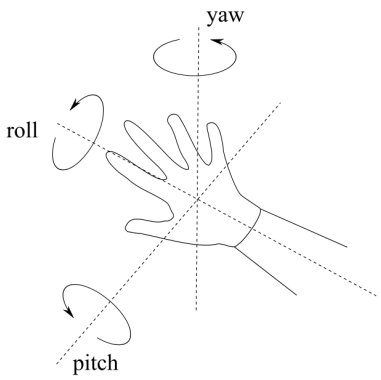


Figure 4: The three dimensional rotation system was first used to describe aircraft principal rotational axes pitch, yaw and roll which can also be designated as lateral, vertical and longitudinal.

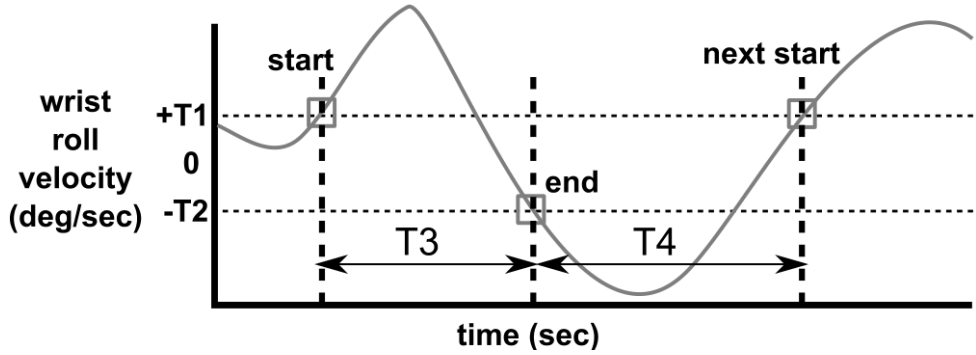


Figure 5: This graphic represents data samples from the roll axis of a gyroscope situated the wrist of a human being while eating an apple. The pattern is easily recognisable and can be used for bite counting and food intake activity recognition. This illustration and the one found in Figure 4 are taken from [1].

2.4 Omnivision OV7670 digital camera data protocol

Similar to a VGA interface [15], the OV7670 digital camera has an 8-bit parallel data output synchronized with an output pixel clock [16]. It also sends active-low vertical (VSYNC or change of line) and horizontal (HREF or change of frame) synchronization signals. Moreover, it supports three image resolutions (VGA, QVGA and QQVGA) and different formats (RGB565, YUV and RAW) which can be configured by the SCCB protocol which operates in a similar way as an I2C interface.

Starting from the upper left corner of the frame taken by the camera, the first vertical line or row is output pixel by pixel, with 16bits for each pixel. Thus, each pixel data is sent in two output pixel clocks with the 8-bit parallel output which most significant bit is analysed in Figure 6. Once the data for the first row is sent, VSYNC is switched to low logical level for a certain delay time and no data is sent. When VSYNC is at again at high logical level, the next row is sent. As shown in Figure 7, each row sending takes around 161μs and the delay for the nest row sending

is about 36μs with 4MHz clock. When all the picture data sending is done, HREF is switched to low logical level for a certain time and then switched back to high level, meaning that the camera is ready to send the next frame as shown in Figure 8.

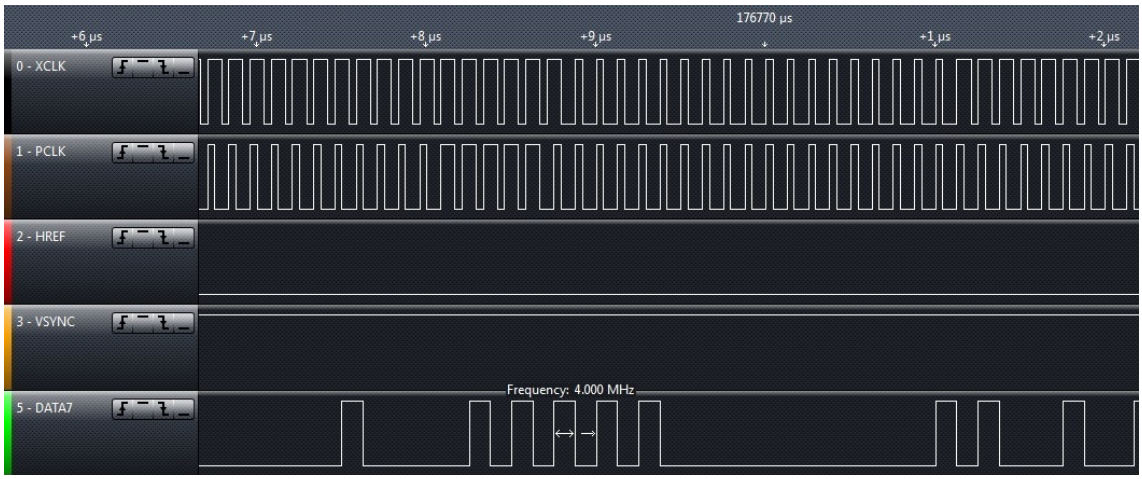


Figure 6: This illustration shows the data measured from a logic analyser with 24MHz of sampling rate. XCLK is the external digital camera input clock at 4MHz and PCLK is the clock generated from the camera, synchronized with the pixel data sending, also at 4MHz. At every PCLK clock, new 8-bit data is sent as can be seen in DATA7, which is the most significant bit of the 8-bit parallel data output. Please note that sampling 4MHz signals at 24MHz leads to low resolution in the sampled signals, this is the reason of why the duty cycle of the clocks appears not to be 50%, but in reality it is.

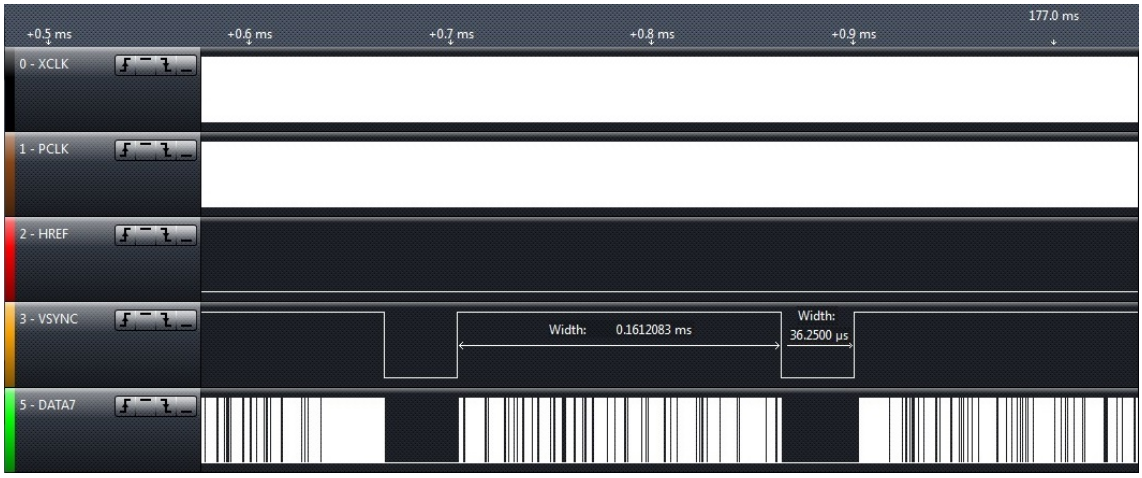


Figure 7: The reader can notice that data is sent only when VSYNC is high. With 4MHz clock, it takes 0.1612083ms±0.1% to send each row. Thus, around 644.8332±0.1% 8-bits are sent and, with 16-bit per pixels, this means that the row resolution is 320 pixels and therefore the resolution is 320x240 pixels.

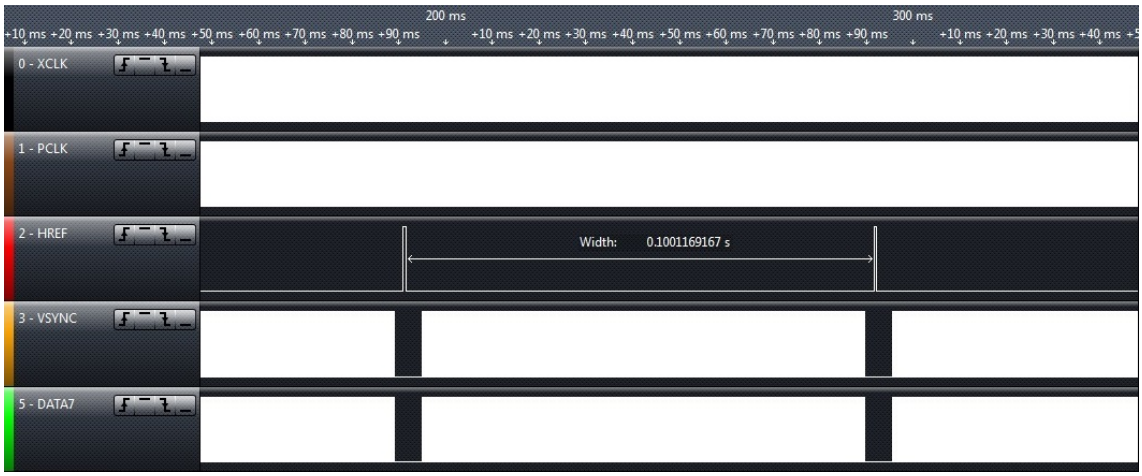


Figure 8: This is an overview of the signal operation of the OV7670 digital camera. Each frame stars with the fall of the signal HREF and finishes with the rise of the same signal. According to the digital analyser, it takes around 100ms to send a frame at 4MHz. This is equal to 10fps and in this way with input clock of 24MHz the frame rate will be around 30fps. The image resolution is QVGA (320x240 pixels) as explained in figure 7.

For a VGA resolution (640x480 pixels), the time needed to send a frame is calculated with (1).

$$Tframe = 2 * 640 * 480 + 639 * tVSYNC\_DELAY + tHREF\_DELAY \quad (1)$$

where tVSYNC_DELAY is stop the time between the last pixel data of a row and the first pixel data of the next row and tHREF_DELAY is the delay between the last pixel of a frame and the first pixel of the next frame.

2.5 Digital image processing applied to food detection and recognition

Traditional food intake monitoring techniques are based on manually responses to questionnaires by the subject. This requieres user intervention and at the same time leads to subjective data [12]. One solution is to monitor the nutritional intake by taking a photo of the meal and apply digital processing and machine learning algorithms. Thus, no human invervention is requiered and the data is automatically sent to the could for further processing.

In computer vision, object detection and recognition can be implemented by several methods, especially when applying classifiers like in [17] or [18]. Feature matching at the base of many computer vision problems, such as object recognition [19]. The SIFT keypoint detector and descriptor has proved to be reliable but it requires too much processing power which is not available on mobile devices. SURF and SIFT algorithms offer a feature matching with lower computational requirements but they are still not capable of real-time processing in mobile devices

[20]. Nevertheless, the object classifier can be implemented on a server by sending the image from the mobile device to the cloud.

Computer vision implemented on a mobile device must be implemented by low-computational power functions [21] such as colour detection and segmentation, canny edge detection or simple filters or morphological operators. The proper combination of these algorithms will lead to a proper object segmentation and contour and other shape related matching can then be applied in order to estimate the type of object.

One solution is better for real-time processing and the other has better performance. Therefore, once the picture is taken, the mobile can use one of the solutions for a real-time recognition and at the same time send the picture to the server where the feature matching algorithm will confirm the object type.

Finally, neural networks are complex solution but with high detection rate. They are one of the best options when the computational power is not a limitation [22].

2.6 Optical heart rate sensor design

The sensitivity of the already existing optical sensors allow it to be located on the wrist, either the upper side or the down side. This location of the down side of the wrist is an advantage in the sense that the product would be smaller on the upper side but it is translated with using wires which will pass through the strap of the wristband. This may lead to bad connection, easy to brake device and therefore lower long-term operating time. It is therefore a good idea to look for alternative designs of optical sensor.

The measurement technique of this of sensor is called photoplethysmography, which relies on light source and optical detector. The most common light wavelengths used by this type of sensors are the infrared, red colour and green colour [4]. The basic operating principle is that the light from the source is reflected more or less by the human being tissue depending on the pressure between the device and the skin. This pressure is related to the blood pressure and in this way the heart rate is estimated.

The simplest design example of this kind of device is shown in Figure 9, where LED is used as light source and the reflected spectra by the skin is sense by a photodiode. The signal conditioning is based on low-voltage operational amplifier followed by a low-pass filter [23].

Moreover, Figure 10 illustrates the block diagram of another possible heart rate sensor design [4]. The light source is a LED that generates periodic square signal at 10 kHz frequency in order to reduce the interference from external light sources. The brightness control block calculates the medium brightness and generates a digital signal that controls the intensity of the light emitted by the source, depending on the received and conditioned signal from the optical sensor (photodiode, PD). The signal conditioning of the reflected signal detected by the sensor is based on a transimpedance amplifier, a bandpass filter and an analog-to-digital converter. Finally, the conditioned signal is demodulated.

More advanced methods based on photoplethysmography biosensors include logarithmic DAC, zero-crossing detectors and non-uniform quantisizers [24], but this

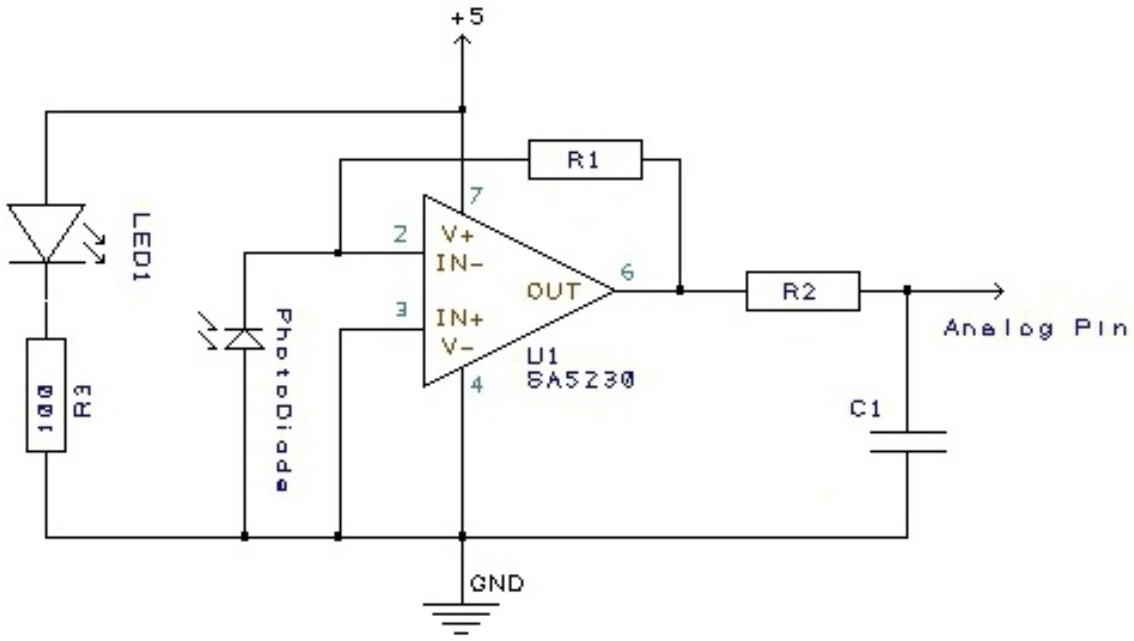


Figure 9: Simple optical heart rate sensor schematic.

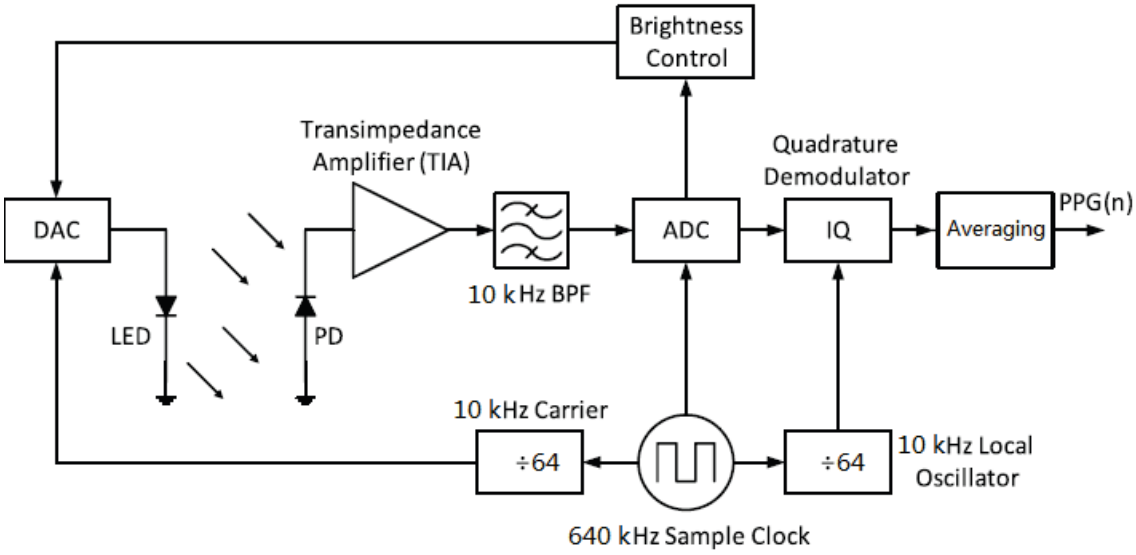


Figure 10: Simple optical heart rate sensor schematic.

raises the size and the time for design and fabrication of the final product.

2.7 Flexible electronic circuit fabrication methods related to wearable devices

Printed electronics are an alternative way to create electronic circuits with the possibility of using a flexible and/or transparent substrate, as illustrated in Figure 11. They can also be combined with silicon chips in order to form a complex electronics

devices [25]. They can be fabricated by different techniques, such as photolithography, ink printing, gravure printing or by roll-to-toll technology.

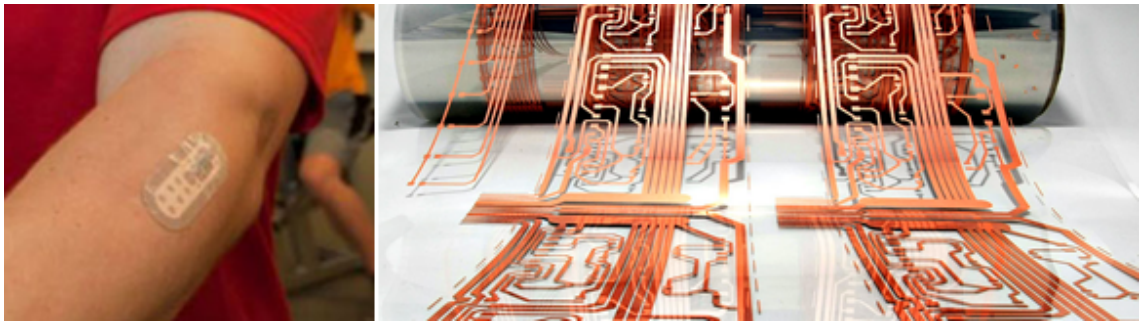


Figure 11: On the left, an example of electronic circuit on a flexible substrate and how it is applied for biomedical purposes. On the right, how the circuit can be created by the gravure imprinting technique.

On the one hand, the gravure printing is high volume manufacturing method [26] where two cylinders are used, one for the impression and other for the gravure. It is a very reliable and cost-effective method for mass production. On the other hand, the inkjet printing operates by a similar way as a normal printer does. Lasers are used to create the vias and special conductive ink is then used to imprint the desired circuit, making it a fast way for prototype with the disadvantage that the conductive ink are usually expensive [27]. Moreover, FPC (flexible printed circuits) are made by photolithography and are mature technology offered by several manufacturers at a reasonable price.

2.8 Summary

In this chapter, relevant research with similar goals was presented, being very useful for the project planning the target setting. It was challenging to find related work that has goals similar to the PRECIOUS wearable design. There is a lot of work about food intake detection from a digital image, but using a bracelet with integrated camera for that purpose is not very common nowadays.

Moreover, the VIM was presented and some of the algorithms to fulfill its parameter have been explained, especially the physical activity tracker and the nutritional intake. Some technical considerations have been presented (the digital camera protocol and the optical heart-rate sensor electrical schematic) which are fundamental for the design of the wearable. Flexible printed circuits were presented as a future development of the wearable, once the prototype is fabricated, programmed and verified properly.

3 Smartphone application development

The PRECIOUS project aim is to develop health related solutions for everyone. For that reason, an Android application is developed as it could be used by every smartphone user. This is also very important for the smart wristband device design because by using a smartwatch it is possible to set up the use cases and to deal with issues that can appear after designing the wristband. It is a very critical step that reveals what can be expected from the smart wristband and what problems can happen, before they occur. Before starting the design and implementation of a prototype and then fully operation device, the Samsung Galaxy Gear smartwatch has been used for testing an Android application, looking for use cases and ensuring that the above mentioned sensors will be enough to track subject's physical activity and food intake habits. The PRECIOUS Android application has been developed for that purpose and tested under normal human being environment. The app is also used to interface the PRECIOUS server, sending the collected data and thus contributing to the creation and update of the VIM but it also receives feedback that it is used to motivate the end user and propose healthcare related recommendations based on his or her profile.

Once the Android application operation is tested and the use cases are set up, an Arduino on protoboard prototype is designed for proving the basic operating principle of the final smart wristband design. Arduino nano board is used to control the MPU-6050 6-axis accelerometer and gyroscope sensor, as well as the OV7670 digital camera, a pulse sensor and the HC-06 Bluetooth module as demonstrated in the next chapter.

The main screen of the application shows information about the physical activity tracking and enables user to load the food intake detection or the face detection programs, as shown in Figure 12. The weight estimation from the face recognition is explained in Section 3.2. Furthermore, the user is able to record the food intake by taking photo of the food, scanning the barcode of a product or just manually choose the food from a database, as described in Section 3.3.

3.1 Physical activity tracker

The Physical activity monitor is a service based on the algorithm explained in Section 2.3 that detects and recognizes the following biometrics:

- Number of steps
- Walking activity
- Running activity
- Riding a bicycle activity
- Travelling in a vehicle
- Using the Android device

Figure 12: On the left, The PRECIOUS Project Android application running on Samsung Galaxy Gear smartwatch. On the right, the main screen of the application tested on the smartwatch.

– Sleep time estimation.

After testing with a smartphone, the collected data was sufficiently accurate. The step counter showed less than 5% deviation from the real results and the estimation of walking, running and riding a bike physical activity time showed deviation of less than 10%.

3.2 Weight estimation

The proposed weight estimation algorithm is based on the principle that the face width of a person at a specific time varies from the face width of the same person at different time if he or she has lost or gained weight, while the face height is maintained constant. Thus, the weight estimation algorithm asks the subject to take a photo of his or her face every week and then compares the face width vs height ratio in order to estimate if the user has gained or lost weight. This is demonstrated in Figure 13.

After taking a picture, face recognition is done using algorithm explained in [28] in order to determine the exact location of the face. After that, the face size is estimated in pixels and the width vs height ratio is calculated and stored.

3.3 Food intake monitoring

The food detection and recognition system offers three operating alternatives. The first one takes a photo from the Android device or wristband camera, recognizes automatically the food and asks the subject for confirmation. The second method relies on a camera barcode scanner and online database product nutritional information. The last one allows the user to manual input the food intake with the help of a

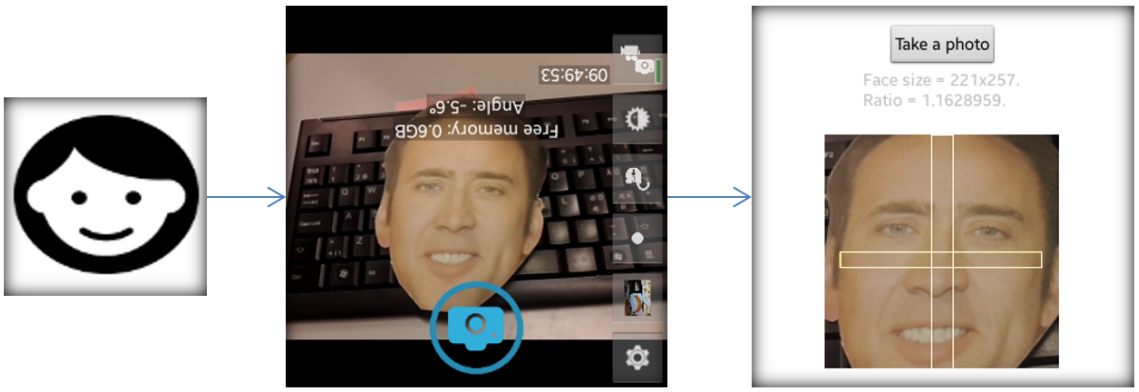


Figure 13: Demonstration of how the face size detection and recognition operates.

database stored in the Android mobile device. The first two methods are illustrated in Figure 14.

3.4 Food image detection and recognition

Computer vision algorithms offer the possibility to automatically detect food and recognize its type. The food intake detection and recognition is very important because once linked to a database it can offer the nutritional information of the food eaten by the subject. With this information, it is possible for example to detect if a diabetic person is not following a diet properly based on the VIM parameters and thus provide feedback and recommendations based on machine learning algorithms running in the server. The algorithm is demonstrated in Figure 15 and Figure 16 and all the steps are then explained. The food image detection and recognition is done by digital image processing using the OpenCV library. The algorithms are first written in C++ and tested on a desktop computer and after that the code has been translated in Java in order to be compatible with an Android smartphone.

3.4.1 Applying Automatic colour balance

White balance is a process that removes unrealistic colour casts which mostly appear when the camera is not able to detect the color temperature of the light source. Furthermore, Automatic White Balance (AWB) is a key issue for cameras to achieve high quality image. It is a very important step that has to be done before the colour filtering [29].

The AWB algorithms proposed here scales the histograms of each colour component of a RGB image in order to achieve a value range of 0 to 255, meaning that certain ratio of the darkest pixels is saturated to black colour and the same ratio of lightest pixels is saturated to white colour.

The implementation of the algorithm is as follows [30]:

– Take the input image, convert colour space in RGB, make a copy and split the three colour channels in separate matrices.

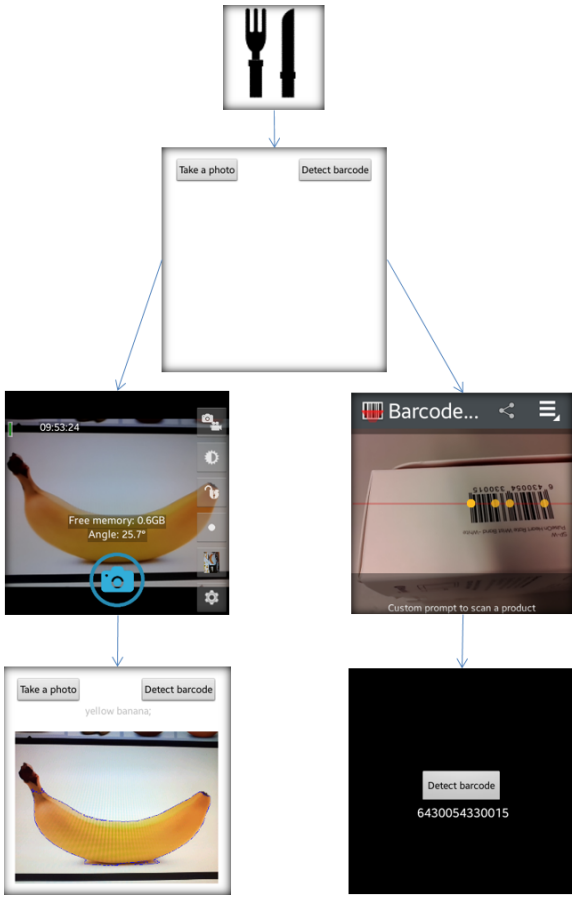


Figure 14: Demonstration of how the food intake detection and recognition operates. The user can choose between taking a photo of a real food, using a barcode scanner or manually inputting the food name.

- Convert each matrix into a vector, sort each vector values from lowest to highest and look for quantiles with probability 0.5% and 0.95%.
- Look for low and high saturation values in each vector according to the location of the quantiles.
- Saturation values and their location is now known. Take the input image in RGB colour space, split to three matrices and apply saturation for every matrix. Finally, merge the three colour component in order to form the automatic white balanced RGB image.

A demonstration of the successful operation of the algorithm is shown in Figure 17.

3.4.2 Colour filtering

In this document, colour filtering means the filtering of a specific colour in a digital image in order to improve the segmentation of object and recognize them in bases of their colour and shape. There are many ways to implement such algorithm.

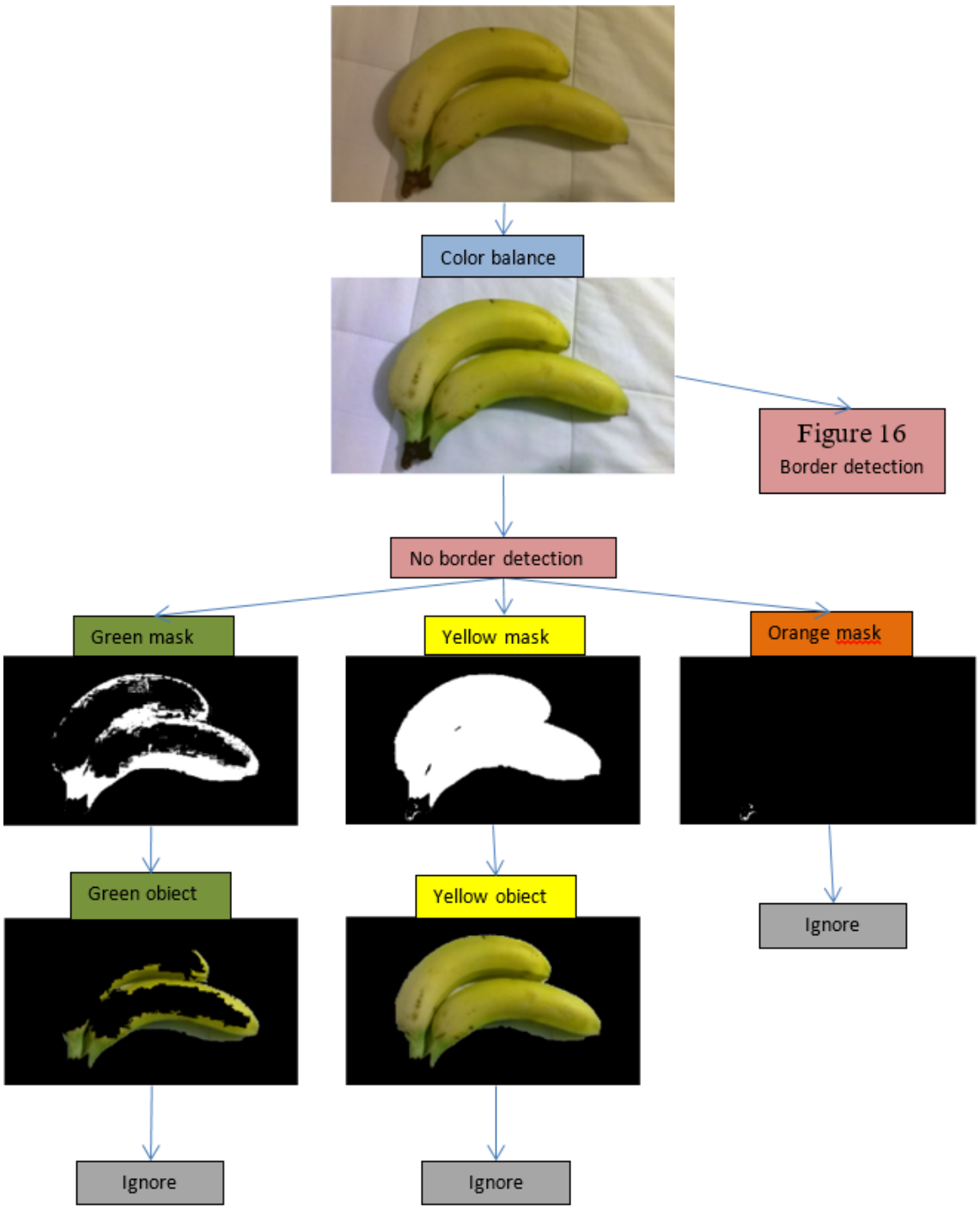


Figure 15: Demonstration of the food intake detection and recognition algorithm, part 1.

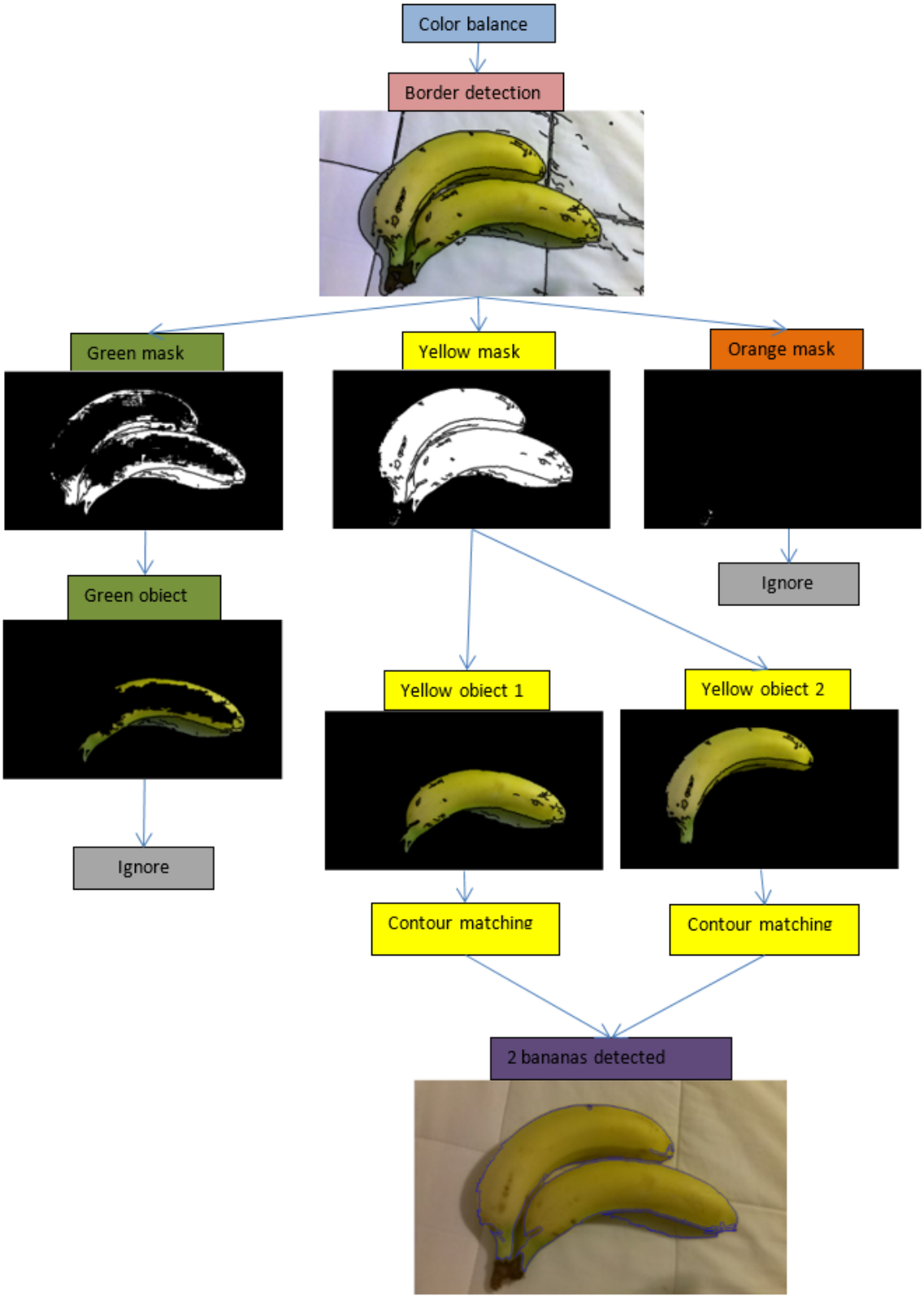


Figure 16: Demonstration of the food intake detection and recognition algorithm, part 2.

Figure 17: On the left, photo of two bananas on white background taken on artificial light environment with poor white balance. On the right, the result of applying the AWB algorithm of the photo on the left.

3.4.2.1 Choosing the right colour space for colour filtering

Usually a digital image is defined by three matrices, each with the size corresponding to the image height and width measured in pixels. Starting from the colour space of the digital image, HSV space is the most common when working with colours [31]. In this colour space, each pixel in the image contains information about the amount of hue, saturation and lightness[1] which means that each of the three matrices that compose the image contains one those three values. On the other hand, the most typical digital image colour space is the RGB, where each pixel contains information about the amount of red, green and blue. A comparison of how these three components are linked to the color perception as illustrated in Figure 18.

Usually the HSV colour space is recommended when colour operations are needed because of the fact that the hue attribute is a digital analog of the physical dominant light wavelength. This means that if a picture is formed by three matrices, each containing hue, saturation and value, the hue matrix contains almost sufficient information about the light wavelength of each pixel and therefore the colour perception can be extracted. An example of green colour filtering applied on a colour map image is shown in Figure 19 where the attribute range are set as follows: hue [38, 75], saturation [140, 255] and value [60, 255].

The main advantage of this technique is that the filtering can be done by changing the range of the hue attribute only, while maintaining the saturation and value ranges constant which will prevent too dark or too light colours to pass through the filter. It is a very effective way to filter a colour spectrum, but it leads to a noticeable issue: there is no specific range for a specific colour. In other words, back to Figure 19, for the human eye the colour depends not only on the dominant wavelength but also on the saturation and brightness of the colour in a way that in a 2D colour map a specific colour (green for example) is not determined by a rectangle area but by a

[1]Lightness is also known as intensity or darkness.

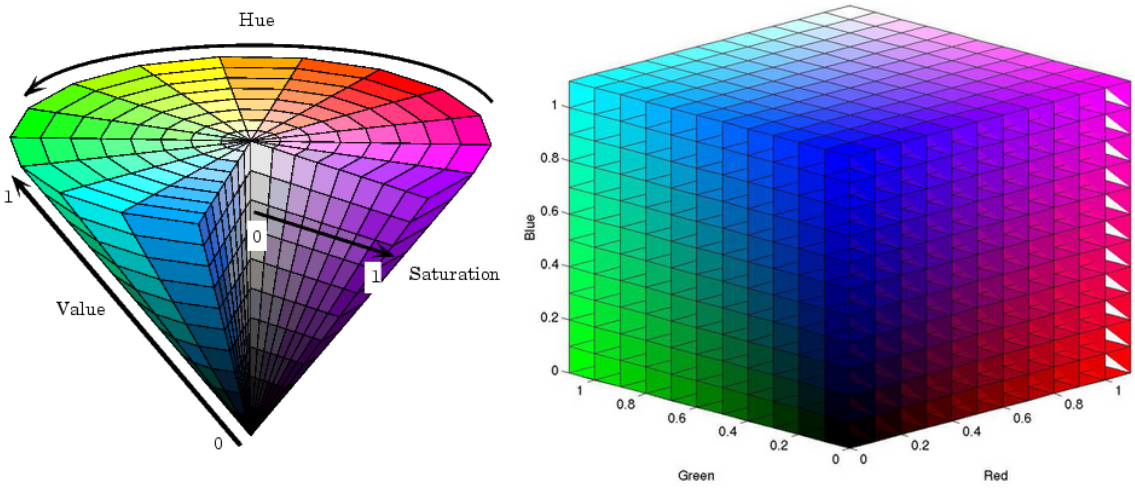


Figure 18: On the left illustration is shown how the components of the HSV colour space are linked to the human colour perception. On the right illustration, the same is done for the RGB colour space.

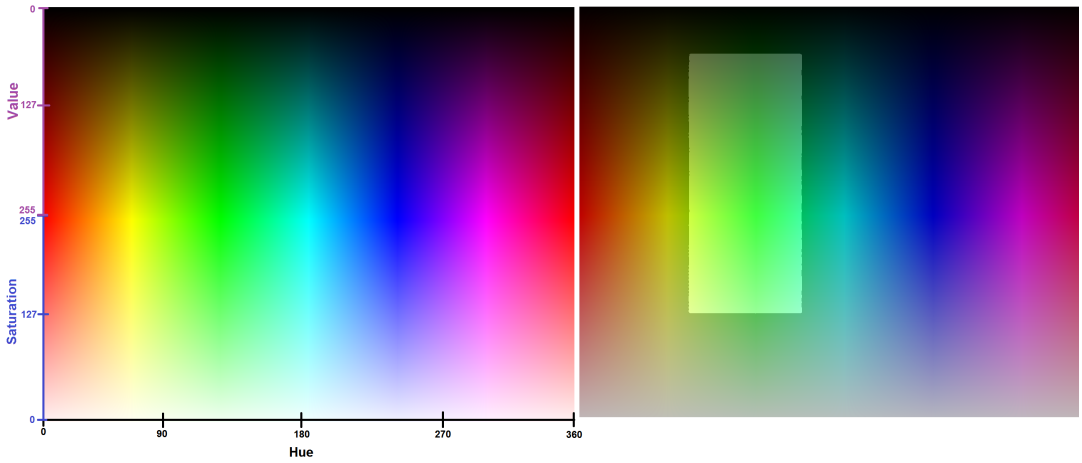


Figure 19: On the left, the HSV colour space attributes are represented on a 2D colour map. On the right, an example of green colour filter is presented where the filtered colour is marked by a white-transparent rectangle. The ranges of the filter are hue [38, 75], saturation [140, 255] and value [60, 255].

triangle area. There is no way to obtain a triangle area using a simple HSV attribute ranges. However, this is possible with the RGB colour space by using a combination of three different ranges for the red, green and blue attribute for each colour wanted to be filtered. Such filtering is illustrated in Figure 20 where the attribute range are red [0,100], green [100,255] and blue [0,100]. Thus, the RGB colour space offers better colour spectrum approximation to the human being perception, but requires to process three attributes instead of one.

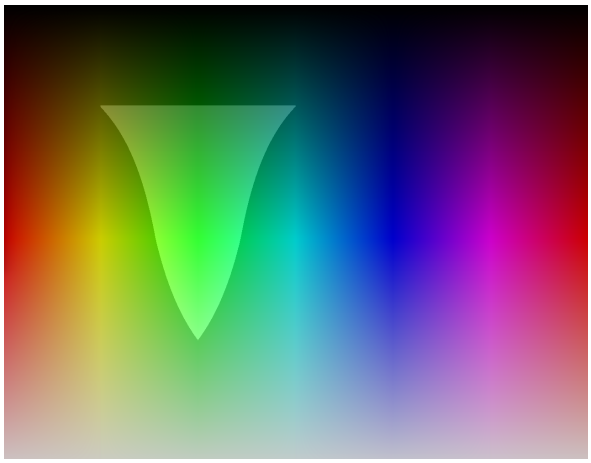


Figure 20: Example of green colour spectrum filtering using the RGB colour space and attribute ranges of red [0,100], green [100,255] and blue [0,100]. The filtered colour is marked by a white-transparent approximation of a triangle.

3.4.2.2 Applying filtering based on object colour spectrum

Even the simplest radio receiver needs to filter most of the non-desired captured signals. This can be done by a band-pass filter where the desired signal is found in the area with a range of [f_C - BW/2, f_C + BW/2], where f_C is the central frequency and BW is the bandwidth of the filter. Every colour has an associated wavelength (and therefore frequency) which is linked to the hue component of a digital image in the HSV colour space. Let the histogram of the number of pixels on the image for each hue value be the amplitude of the input signal and the hue value be the frequency. It is therefore theoretically possible to choose a central frequency (or central hue value) and bandwidth in order to filter an object with a specific but not uniform colour. This method is called *object colour spectrum filtering* in this document.

For example, if a green objects are to be detected, after the basic HSV filtering shown in Figure 19, most of the non-green objects will be filtered but there still be a noise coming from the yellow-green and the green-blue objects. This noise cannot be removed because the exact colour range of the objects to be detected is unknown. However, the histogram of the hue component of the filtered image can be used as an input signal of the object colour spectrum filter, being the central frequency the peak of the histogram and the BW will be calculated, for example, from the -3dB (or 70.7% of the maximum amplitude) hue value fall.

For better understanding, on Figure 21 is shown the histogram of the hue component after applying a basic HSV filter with range hue [38, 75] on the green apple. The object colour spectrum filter will have a central frequency of hue=53 with 449 pixels of maximum amplitude. Let the BW be at 20% of the maximum amplitude, which is 90 pixels, resulting in a BW of [45, 59]. Filtering the hue component with range [45, 59] instead of the initial [38, 75] removes the unwanted noise which in this particular case is caused by the reflection of the green apple on the surface.

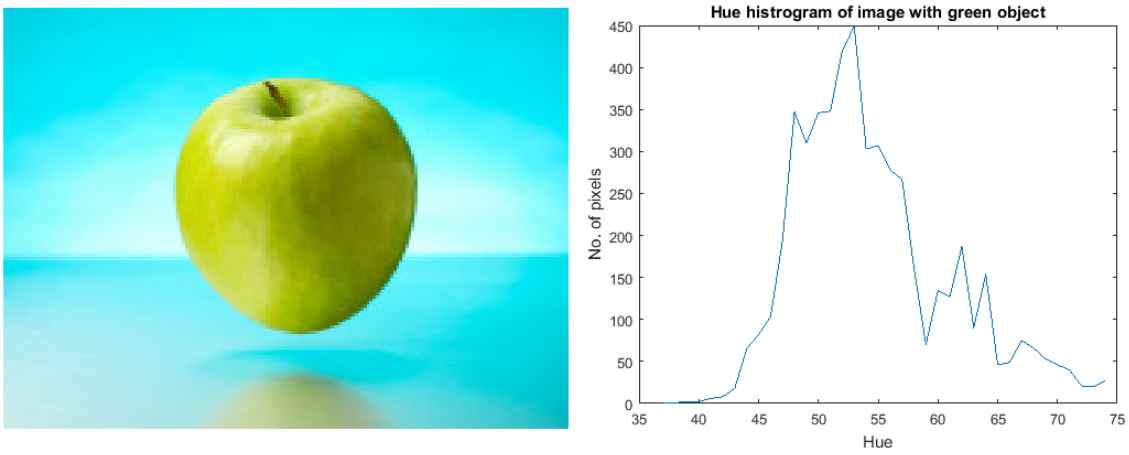


Figure 21: This illustration is an example of how the input signal for the object colour spectrum filter looks like. After applying simple HSV colour filtering with range hue [38, 75] (corresponded to the green wavelength) on the image on the left, the histogram of the hue attribute is calculated and plotted in the figure on the right. This plot is useful to understand the analogy between the input signal of a radio receiver and the colour spectrum of the image. This analogy is the basic principle of the object colour spectrum filter.

3.4.3 Object segmentation

The colour detection and filtering is the first step of the object segmentation and sometime is enough to detect the desired object, as demonstrated in Figure 22. Nevertheless, once the estimated colour of the object is detected, there must be an algorithm that ensures that there is only one object detected and, if there is more than one, it must be capable of separating them. This is achievable with a canny edge detector, which detects the edges of the object, and morphological operators such as dilation and erosion for a proper separation of the objects. For example, in Figure 23, the colour filter will be unable to estimate properly the colour and all the apples will be detected as one object. However, by applying object segmentation, the apples are easily separated.

3.4.4 Contour detection and pattern matching

After segmentation, the object is properly detected but now it needs to be recognized. The most computational effective algorithm is to find the contours of the object and try to match it with already known object contours from an existing database. This is very easy to be implemented and the results and shown in Figure 24.

3.5 Summary

Smartphone application was developed and it will interact with the wristband, being fundamental for its proper operation. The physical activity and food monitoring algorithms were developed, being the physical activity tracker the same both on the

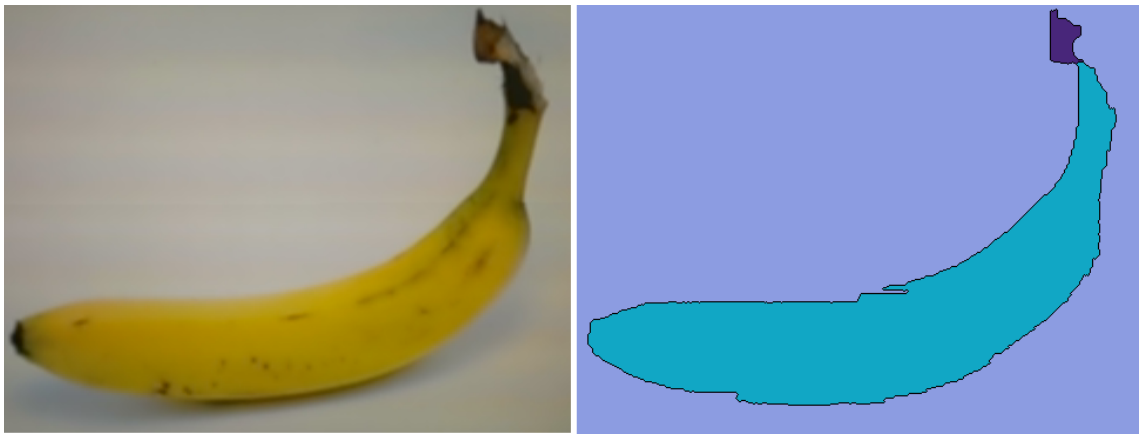


Figure 22: Demonstration of object detection by only applying color filtering

Figure 23: Demonstration of object detection by applying color filtering and object segmentation

smartphone and on the wearable. Thus, the PRECIOUS ecosystem can interact with a smartphone only if needed, being reachable by many subjects. The wristband complements the application and adds some important features like the continuous tracking, sleep quality monitoring and easier nutritional tracking. Nevertheless, the mobile application is fundamental because it is both used for the user interaction and to interface the server responsible for the VIM.

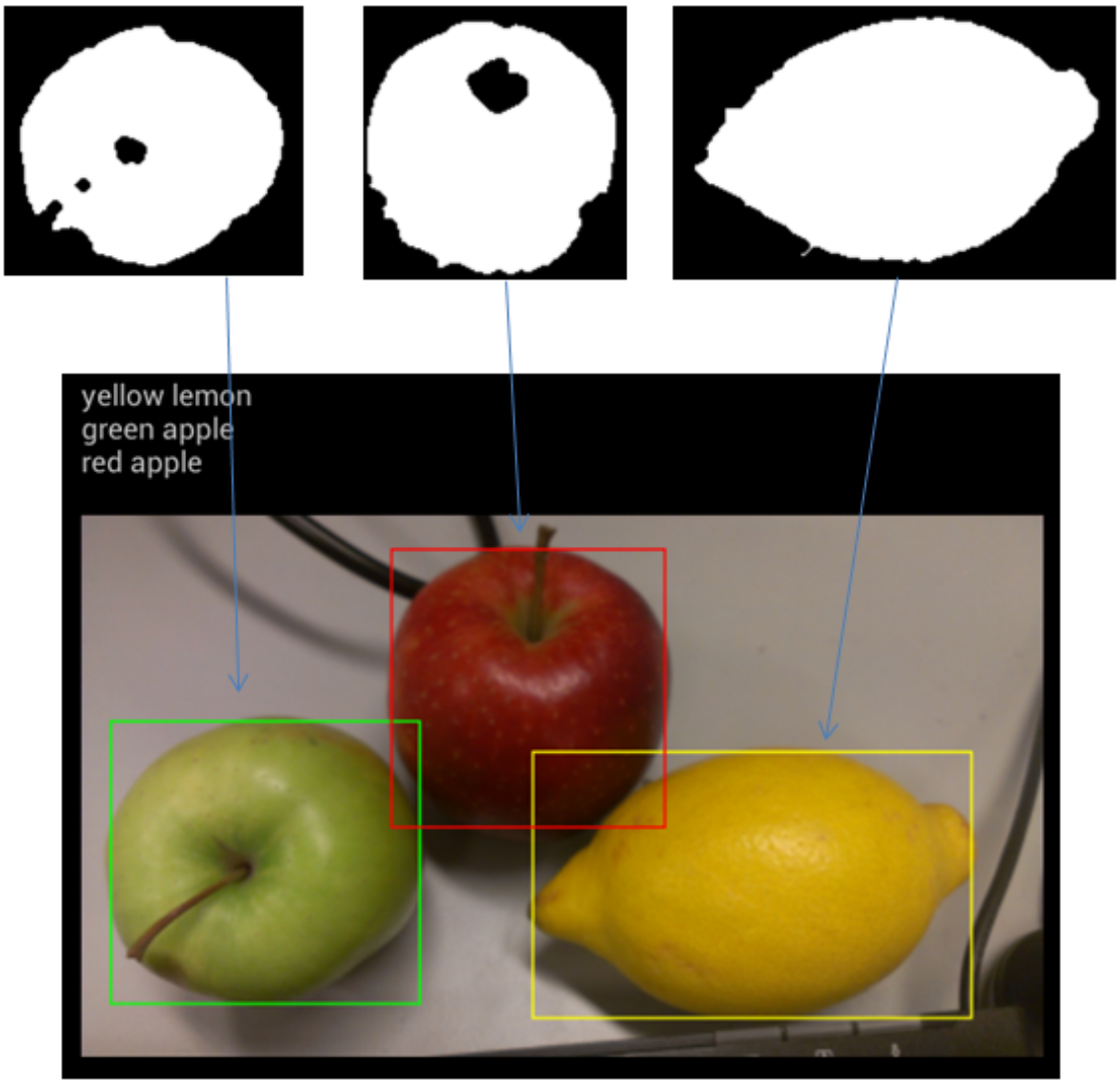


Figure 24: Demonstration of object detection by applying colour filtering and object segmentation

4 Wearable design, fabrication, programming and verification

In this chapter the wristband development is presented, following the most common rules for designing an electronic product. First, the system is designed and all the components are selected. After that, the operation of each module and then of the whole system is tested on a prototype board. The device is then designed in a CAD environment and fabricated on a PCB by two different techniques. Finally, the wristband electrical connections are verified and, after programming, the correct operation of the wearable was proves.

It is very important to remember that the data collected by the wristband is sent to the user's smartphone by Bluetooth and after that it is stored in the cloud, being accessible for the VIM creation and update.

4.1 Preliminary design on a proto-board

The goal of the prototype design is to achieve a basic functionality in order to prove the final device usability. The prototype should be an easy and fast to build device. Arduino microcontroller environment offers support with many devices and peripheries. It is therefore a suitable way to build a first prototype.

However, the final prototype should be more platform-independent and smaller device. Atmega328 microcontroller, used in some of the Arduino devices, offers small SMD encapsulation and is both small and compatible with the Arduino software. It is therefore suitable for the final prototype design.

The Arduino prototype is the first step of the wristband design and it will enable to prove its functionality. Arduino nano board is used to take the control of the sensors and send the data to a mobile device via Bluetooth. Moreover, accelerometer MEMS device consumes less power and chip area in comparison with other sensors and at the same time provides very useful information which can be used to control the power of the rest of the sensors or to estimate walking or running time. Gyroscope MEMS sensor is needed to monitor the wrist movements, recognize food-intake activity and even count the number of bites [1].

Furthermore, a camera sensor will take photo before and after the food-intake activity. All the information will be sent in real-time to a smartphone via Bluetooth. The proper operation of the device will be monitored by using LEDs. Finally, optical heart-rate sensor will monitor the heartbeat.

4.1.1 Integrated modules and other components

In this section, the main built-on modules and other components are listed and their specification and configuration is explained. For the proto-board prototype, Arduino nano board is used because its microcontroller is the same as the PCB prototype, which is the Atmega328P. For the Bluetooth connection, the HC-06 module was chosen because it is a cheap and commonly used solution. For the same reason, the activity sensor is integrated in the MPU6050 IC which is available on a PCB ready

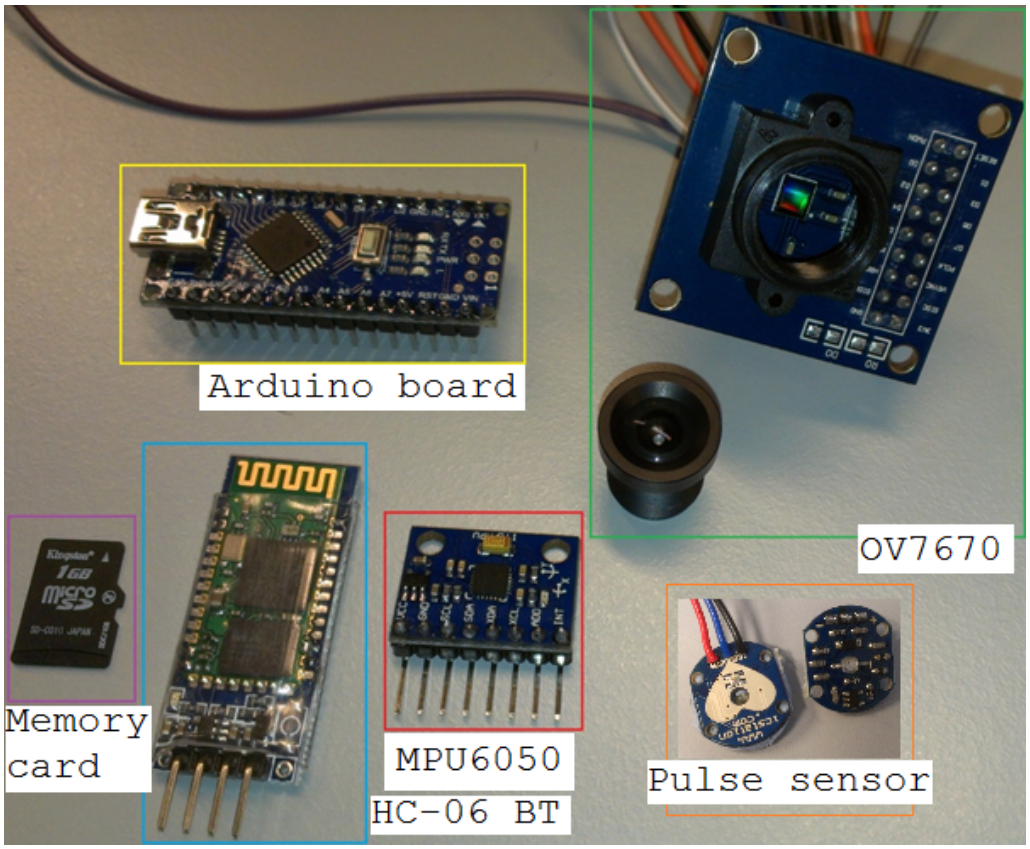


Figure 25: Photo of the main component needed to build the protoype.

for prototyping. The digital camera is the Omnivision OV7670 which is also a cheap but also well documented device. Moreover, micro-SD card is used for data storage. These components are available in Figure 25 and cost estimation has been done in Table 1

Table 1: This table shows a cost estimation of the components need to build the PRECIOUS wearable device.

Component name	Cost in euros
BT module HC06	3.03
MPU-6050 module	2.27
OV7670 FPC	2.64
Pulse sensor	5.23
Atmega 328P SMD	3.15
3.3V 500mA LDO	0.39
5V 500mA LDO	0.8
SMD caps/resists	<1
Total cost in euros	18.51

4.1.2 HC-05 Bluetooth V2.0 SPP module test

The operation of the module was tested under the following set up:

- 3.3V power input
- 9600 baud rate transfer.

The module was tested with a simple program that sends and receives data from the microcontroller via UART, being the Bluetooth communication transparent for the MCU. On the other side of the Bluetooth connection, the Android phone receives and send data to the MCU.

4.1.3 MPU-6050 3-axis gyroscope, 3-axis accelerometer and Digital Motion Processor test

The operation of the module was tested under the following set up:

- 3.3V power input
- I2C communication on port 0x6B[2].

The module was tested with the program available online in [32]. The information from the sensors is received via I2C communication and sent via UART to the PC. An example of the data can be found in Table 2, which proves the correct operation of the device.

Table 2: Example data received from the motion sensors of the MPU-6050 device.

AcX	AcY	AcZ	GyroX	GyroY	GyroZ
16380	-516	2864	-362	264	50
16380	-516	2864	-362	264	50
16520	-488	2932	-350	252	51
16296	-420	2984	-350	260	52
16564	-436	2964	-352	250	55
16400	-468	2856	-367	246	58

4.1.4 Omnivision OV7670 CMOS VGA camera module test

The operation of the camera was tested under the set up shown in Table 3. The Arduino was programmed to receive the 8-bit parallel input image data according to the standard protocol used by the OV7670 camera module explained in Section 2.4 and send it via UART interface; the code is available in Appendix A. On the other hand, a Pyhton computer program receives the image from the Arduino via COM serial communication and stores the data into a file; the code is available in Appendix B. Finally, the raw image data is processed in Matlab and saved in png format using the code available in Appendix C. A successful recoded image is available in Figure 26.

[2]The port number was obtained with a port scanner available in [20]

Table 3: Connection between the Arduino Uno board and the OV7670 camera module

Arduino	OV7670	Description
3V3	3V3	VDD
GND	GND	GND
3V3	RESET	Camera reset
GND	PWDN	Power down mode
A5	SOIC	SCCB clock
A4	SOID	SCCB data
D11	XCLK	Camera input clock
D2	PCLK	Camera output clock
D3	VSYNC	Vertical sync signal
D10	HREF	Horizontal sync signal
A0-A3	D0-D3	Lower part of data output
D4-D7	D4-D7	Higher part of data output

Figure 26: Received image from the OV7670 camera module in YUV422 format. Only the Y colour component is shown in this figures.

4.1.5 Heart-rate pulse sensor module test

The operation of the module was tested by supplying 3.3V power input and connection the sensor's output to the A0 analog input of the Arduino. The module was tested with the open-source program available in Github (https://github.com/WorldFamousElectronics/PulseSensor_Amped_Arduino) with some small modifications. The pulse sensor analog output signal and its low-pass filtered plot can be found in Figure 27.

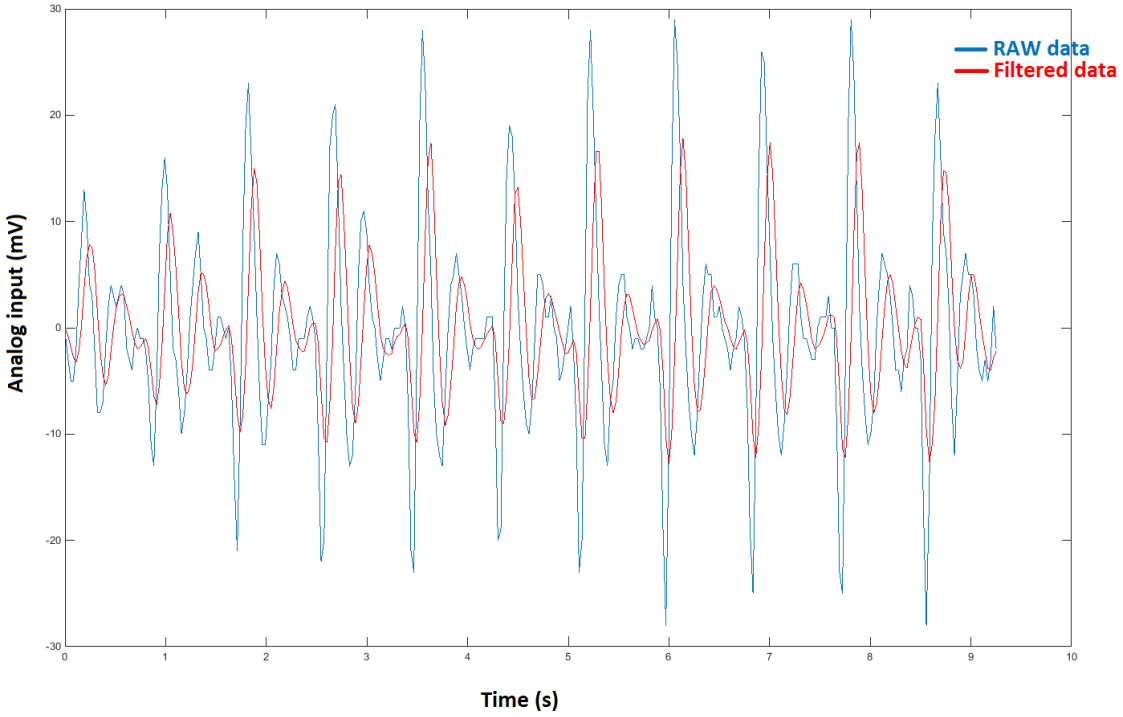


Figure 27: Received image from the OV7670 camera module in YUV422 format. Only the Y colour component is shown in this figures.

Despite the fact that the sensor works properly when sensing data from the forefinger, the sensitivity is not enough to sense data from the upper side of the wrist and therefore the sensor location should be located in the down side of the wrist or another sensor should be found or designed.

4.2 Final prototype design on PCB board

In Section 4.1 was demonstrated the correct operation and successful communication of the microcontroller board Arduino Uno and the other devices. In this section only the heart of the Arduino nano (the microcontroller Atmega328) will be used and connect to all the devices at the same time in order to achieve fully-operational prototype.

The Arduino board was replaced with the following components in order to reduce the size of the final product:

- Atmega328P MCU
- MPU-6050 module in QFN package
- Omnivision OV7670 CMOS VGA camera module in flex package
- Heart-rate "pulse sensor"
- Input power: 3V
- Communication: analog input
- Voltage regulators for 5V, 3.3V and 1.8V.

4.2.1 Atmega328P connection and configuration

Atmega328P is a MCU with 32KB program memory, 2KB RAM memory, 23 I/0 pins (8 of them are analog) and 20MHz of maximum CPU speed. It offers I2C, SPI and UART communication protocols. The minimum and maximum supply voltages are 1.8V (maximum operation frequency of 4MHz) and 5.5V.

An operating frequency of 16MHz offers good performance, suitable for operating with the camera module but it requires at least 3.78V of supply voltage, as shown in the graphic in Figure 28 [33]. Thus, the 3.3V LDO used to supply the camera, MEMS and pulse sensor module is not suitable for the MCU. There it a need to use a separate LDO and 5V of output were chosen because it is a standard value that may be needed in the future to power other modules.

Furthermore, the connection between the microcontroller and the other components has been done following the Table 4[3].

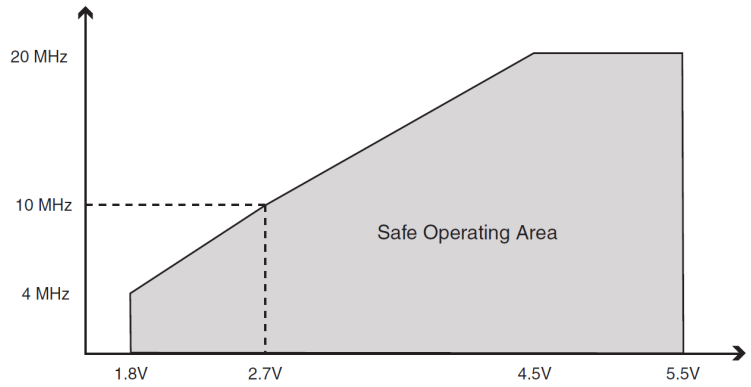


Figure 28: Atmega328P maximum frequency vs supply voltage. By interpolation, the minimum supply voltage for a maximum frequency of 16MHz results in 3.78V

[3]AVCC is connected to ground because the analog signal from the optical heart rate sensor is expected to have a range of 0V to 5V.

Table 4: This table illustrates how the Atmega328P MCU is connected to the other components.

MCU PIN name	Connected to	Description
VCC	5V LDO: Vout	Digital power supply
AVCC	5V LDO: Vout	Analog power supply
GND	5V LDO: GND	Ground
AREF	5V LDO: GND	Analog reference
A1	PULSE: Output	A/D IO, heart-rate data
A0,A2,A3	5V LDO: GND	A/D IO, not used
A5	-	A/D IO,
A6	OV7670: HREF	A/D IO, horizontal sync
A7	OV7670: VSYNC	A/D IO, vertical sync
SDA	MPU-6050: SDA	Data pin of the I2C bus
SCL	MPU-6050: SCL	Clock pin of the I2C bus
RESET	5V LDO: Vout	Reset pin
D0-D2	-	Digital IO, not used
D3	HC06: RX	Digital IO, UART transmit
D4	HC06: TX	Digital IO, UART receive
D5-D11	OV7670: D0-D7	Digital IO, camera data
D12	OV7670: XCLK	Digital IO, camera CLKin
XTAL1,XTAL2	16MHz crystal	External oscillator pins

4.2.2 MPU-6050 MEMS module connection and configuration

MPU-6050 is a 6-axis accelerometer and gyroscope with Digital Motion Processor, which communicates with the microcontroller via I2C bus. On the one hand, the gyroscope offers ranges of ±250, ±500, ±1000 and ±2000°/s with operating current of 3.6mA and standby current of 5μA. On the other hand, the range of the accelerometer is ±2g, ±4g, ±8g and ±16g with operating current of 500μA. This last one has low-power current configuration of 10μA@1.25Hz, 20μA@5Hz, 60μA@20Hz and 110μA@40Hz. The maximum operating current of the MPU-6050 module is 3.9mA, the input power supply range is 2.375V to 3.46V and operation frequency is 32.768 kHz to 19.2 MHz.

The connection of the module is illustrated Table 5, following the instructions of the manufacturer [34][35].

4.2.3 HC-06 Bluetooth module connection and configuration

The HC-06 is a Bluetooth V2.0 SPP module with integrated antenna and UART communication protocol. It can be powered at low voltage from 3.1V to 4.2V and the current consumption is 40mA when pairing and 8mA when communicating. Its RF transmitting power is 4dBm and the sensitivity is -80dBm.

The configuration of the module is already done by the manufacturer and there

Table 5: This table illustrates how the MPU-6050 MEMS chip is connected to the other components.

MEMS PIN name	Connected to	Description
VDD	3V3 LDO: Vout	Power supply
VLOGIC	3V3 LDO: Vout	Digital IO power supply
GND	3V3 LDO: GND	Ground
CLKIN	3V3 LDO: GND	External clock, not used
SDA	ATMEGA: SDA	Data pin of I2C slave bus
SCL	ATMEGA: SCL	Clock pin of I2C slave bus
AUX_DA	unconnected	I2C master data, not used
AUX_CL	unconnected	I2C master clock, not used
AD0	3V3 LDO: GND	LSB of I2C slave address
REGOUT	Cap 100nF	Regulator filter capacitor
FSYNC	3V3 LDO: GND	Frame sync input, not used
INT	unconnected	Interrupt output, not used
CPOUT	Cap 2200pF	Charge pump capacitor

is no need to reconfigure the parameters. Thus, the connection was done as shown in Table 6.

Table 6: This table illustrates how the HC06 Bluetooth module is connected to the other components.

MEMS PIN name	Connected to	Description
VCC	3V3 LDO: Vout	Power supply
GND	3V3 LDO: GND	Ground
TX	ATMEGA: RX	UART transmit
RX	ATMEGA: TX	UART receive
Others	Unconnected	Not used

4.2.4 Power supply management

The prototype is power by battery and voltage conversion is done by 3 LDOs connected as shown in Table 7. The first one is the LM2940IMP-5.0 which outputs 1A of current at 5V and powers the Atmega328P MCU. The second one is the Microchip's MCP1826S-3302E/DB and outputs 1A at 3.3V with minimum supply of 2.3V and maximum at 6V. It is used as power supply for the OV7670 camera module, the MPU-6050 MEMS module, the HC-06 Bluetooth module and the heart rate optical sensor. The third LDO is the MCP1700T-1802E/TT which outputs 250mA at 1.8V with minimum voltage of 2.3V and maximum of 6V; it is used to power the digital logic core of the OV7670 camera module.

Table 7: This table illustrates how the voltage regulators are connected to the power lines.

Battery	LM2940IMP	MCP1826S	MCP1700T	Power line
Vbat	Vin	Vin	Vin	-
GND	GND	GND	GND	GND
-	Vout	-	-	5V
-	-	Vout	-	3V3
-	-	-	Vout	1V8

4.3 Final schematics and PCB layout

The final version of the circuit schematics is available in Figure 29 and the double-sided PCB layout on Figure 30. The CAD software used was Altium Designer v13.3 with Aalto University Licence.

4.4 Fabricating the wearable

The fabrication of the electronic circuit has been done on a FR4 substrate with 250mm thick copper layer. Two different methods have been used. The first one, so called the toner transfer or ironing method, was used for fabricating the circuit in home environment for easier and faster testing of the operation of the device and for testing different ways to solder the most critical device, the MPU-6050 sensor in QFN package. This method offers bad results for processes with line-widths lower than 500mm[4] and for this reason the circuit was fabricated by using photolithography with professional tools in the Design Factory building of Aalto University.

4.4.1 PCB fabrication by the method of toner transfer

The method of toner transfer is a simple home-made PCB fabrication method that avoids using expensive UV-exposure tools. The materials needed are just a PCB board with a copper layer, a laser printer, an iron and glossy paper. The circuit is first printed on the glossy paper and after that the toner is transferred to the copper layer by heating the surface of the glossy paper with the iron. Once the toner transfer is done, the glossy paper is removed and the imperfections and repaired with a proper pen (Figure 31a). After that, the board is etched in copper etchant for 15 to 20 minutes[5](Figure 31b). Finally, the board surface is polished in order to clean the toner and pen residues (Figure 31c). The cleaning step is very important for the soldering of the components.

As a result, it was possible to solder the most critical component, the MPU-6050 chip in QFN package in a way that its operation was successful. However, as

[4]The designed circuit has components with pin width of 200mm with 300mm of pitch.

[5]The copper etching time depends on many factors, such as the concentration of the Cu etchant, the thickness of the Cu layer, the temperature of the etchant, etc.

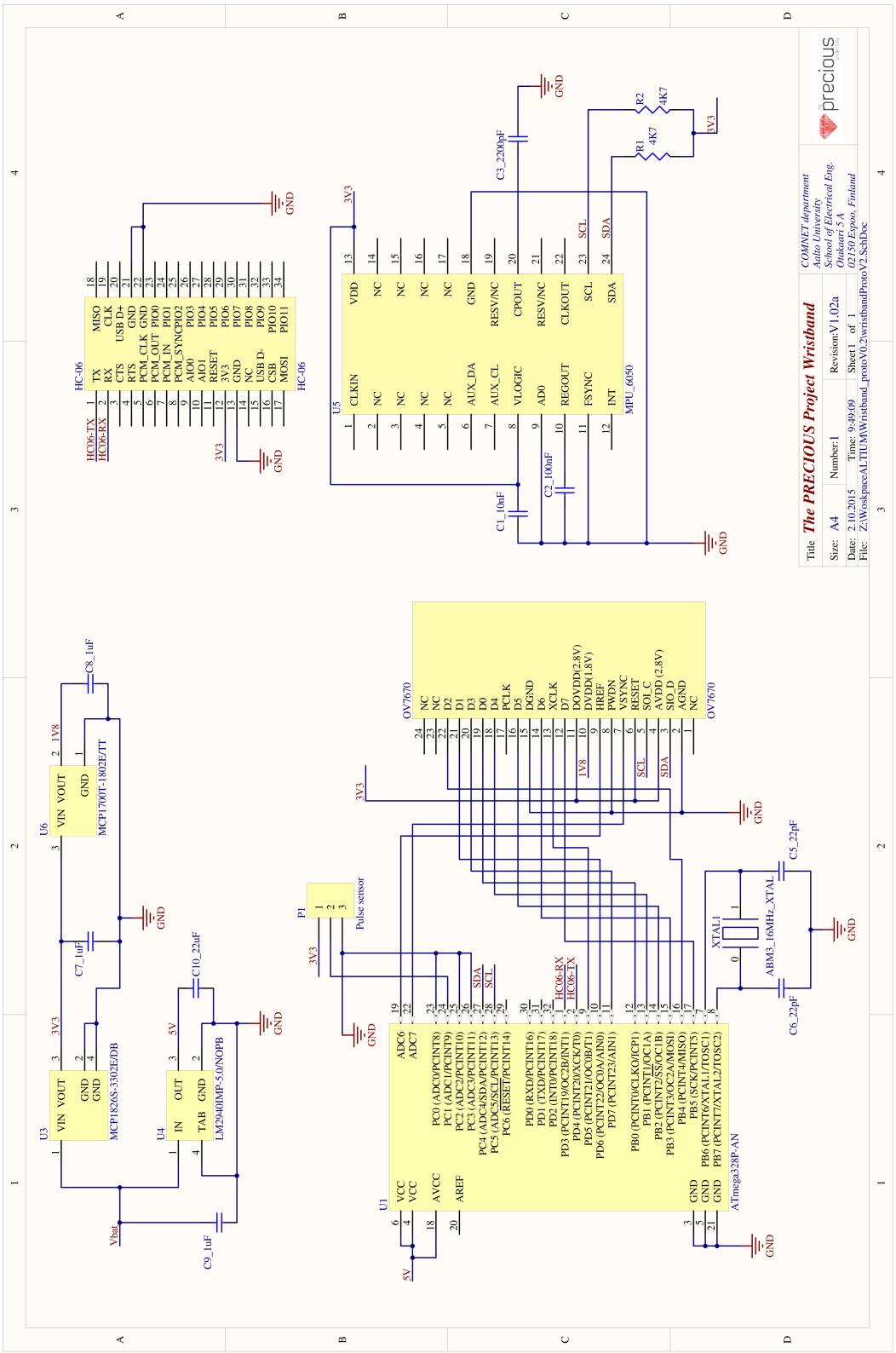


Figure 29: Final version of the smart wristband prototype schematic

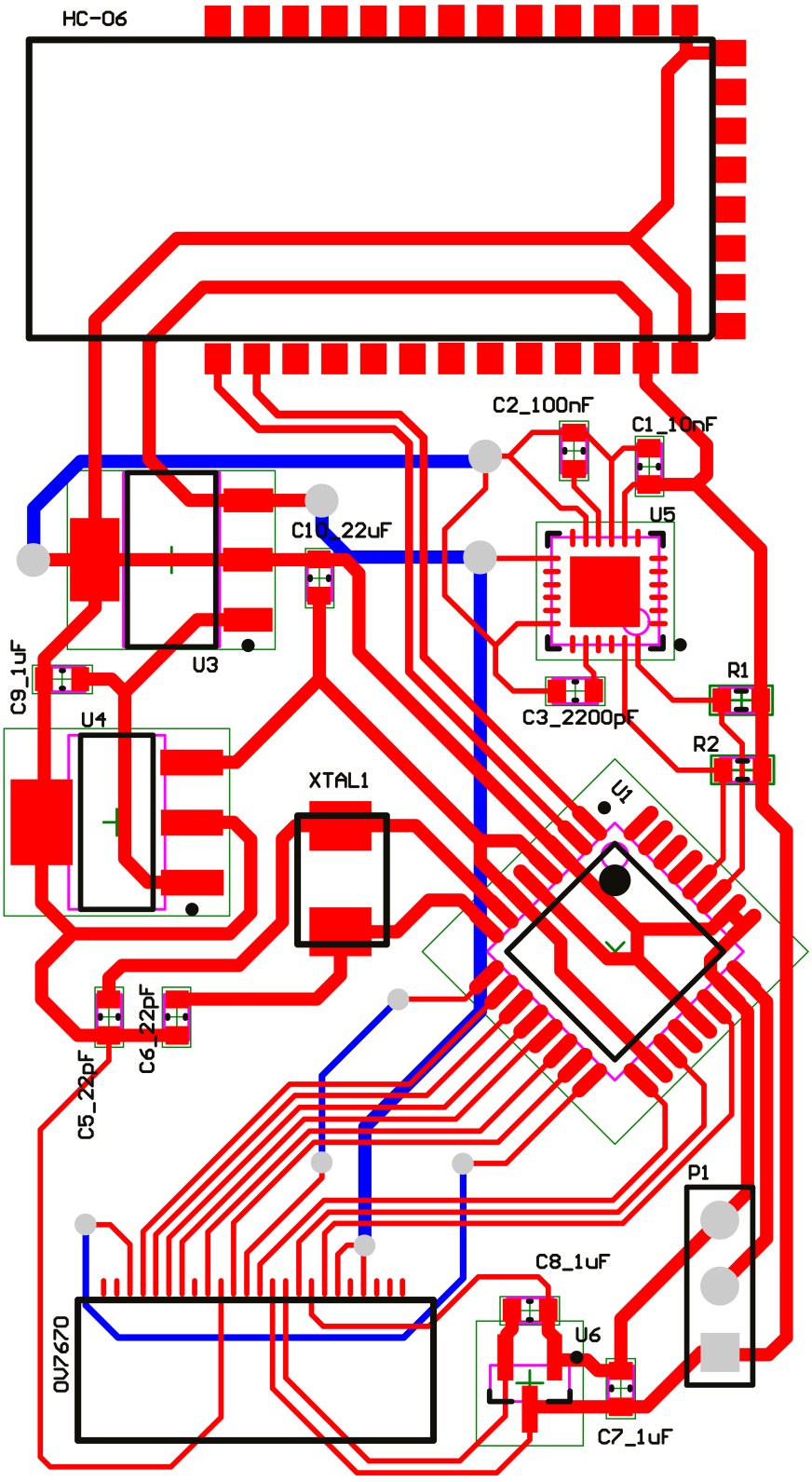
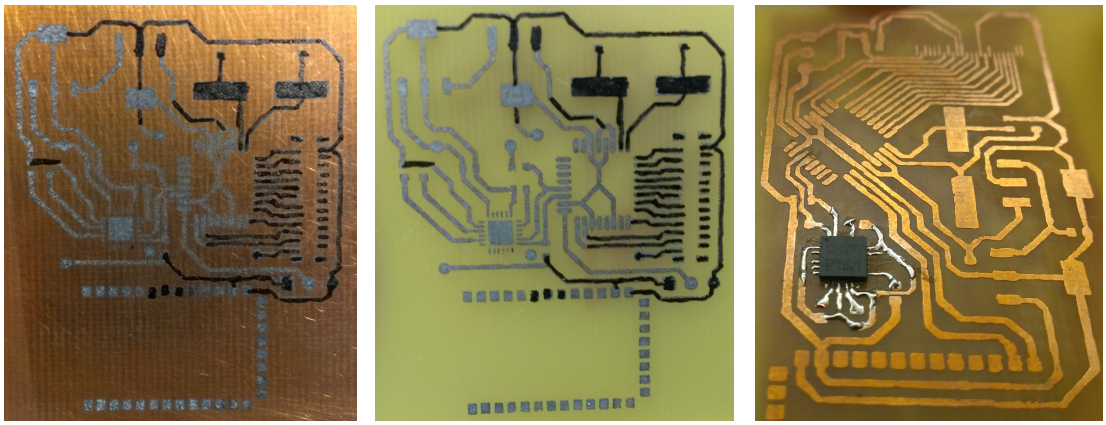


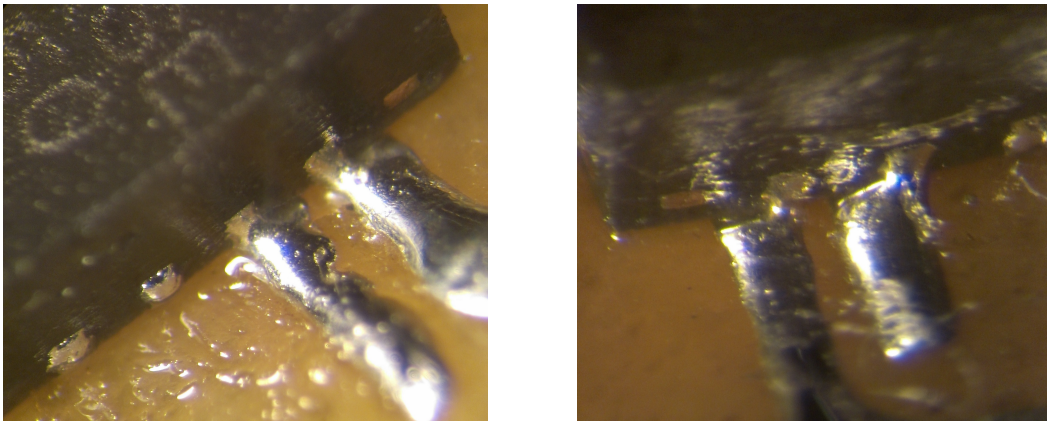
Figure 30: Final version of the smart wristband prototype schematic



(a) After toner transfer (b) After Cu etch (c) After cleaning

Figure 31: This figure illustrates the PCB fabrication steps under home environment, by the toner transfer method.

illustrated in Figure 32, the integrated circuit soldering did not go well because of a misalignment in one of the axis. Despite of it, the connection of the device and the tracks has been checked after soldering and there were no short-circuits, bad connections or unconnected pins.



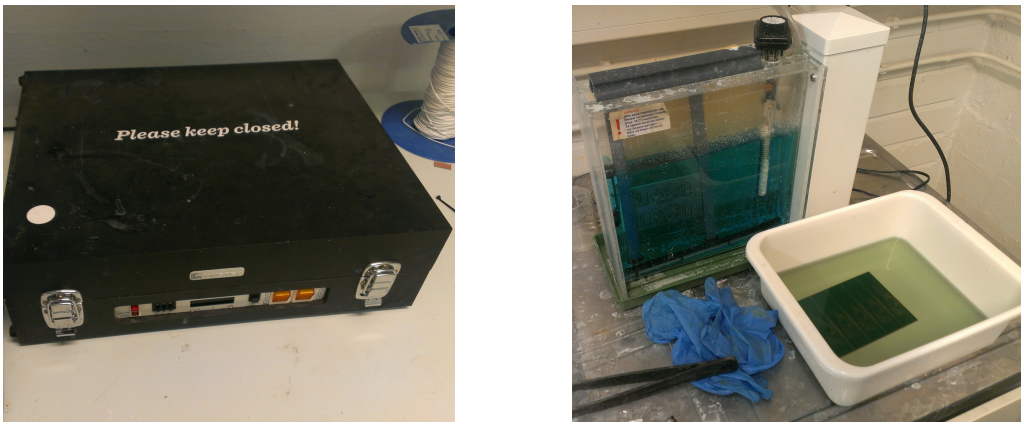
(a) Good vertical alignment (b) Horizontal misalignment

Figure 32: This figure shows how the MPU-6050 MEMS IC was soldered on the homemade PCB. As it can easily be seen in Figure 32b, there is a misalignment in one of the axis, but the alignment in the other axis is good (Figure 32a). The misalignment is most probably due to the bad resolution of the PCB fabrication process, which leads to inaccurate location of the tracks.

4.4.2 PCB fabrication by the method of UV-exposure

Photolithography is key in most of the fabrication steps of PCB boards and silicon chips. It is a professional fabrication method and offers very good results. PCB with copper and photoresist layer is needed as a material and UV-exposure device and

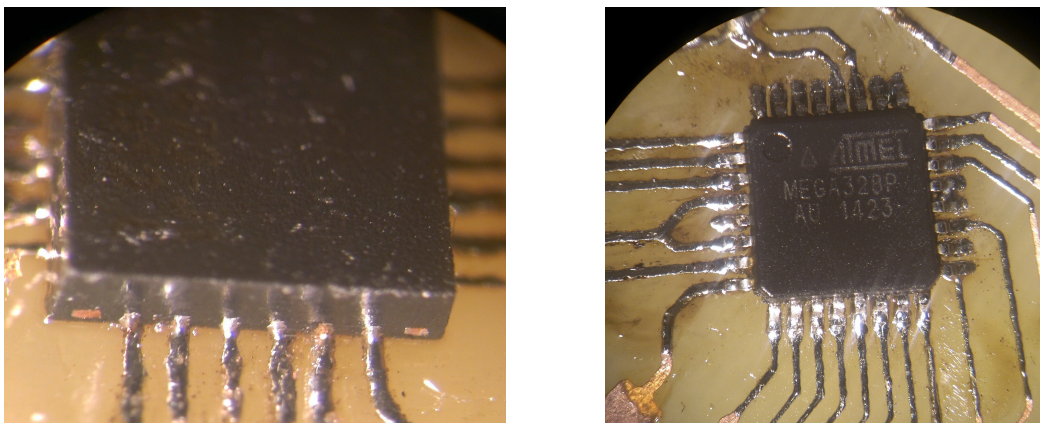
photoresist and copper etching tanks are needed as fabrication tools. The circuit is printed on a transparent foil and after that the patter is transferred to the photoresist layer by UV-light exposure. After that, first the photoresist is etched and after that the copper is etched in the tank. Photos of the fabrication tools are available in Figure 33.



(a) Double side UV-exposure device (b) Photoresist and Cu etching tanks

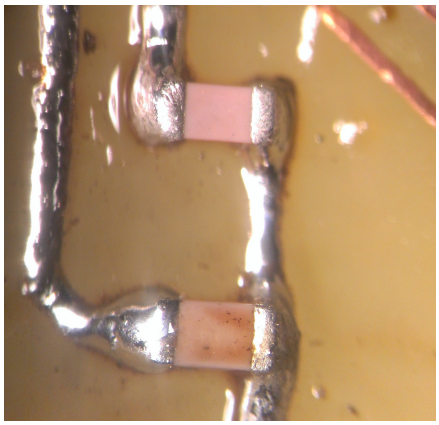
Figure 33: This figure shown photos of the tools used in the photolithography PCB fabrication process.

The fabrication results were with better resolution and accuracy compared to the homemade method and the QFN package was properly soldered on the board with the rest of the components (Figure 36). Furthermore, the soldering of other critical components, like the Atmega328p SMD chip shown in Figure 34 or 0603 SMD capacitors and resistors (Figure 35), was also successful.

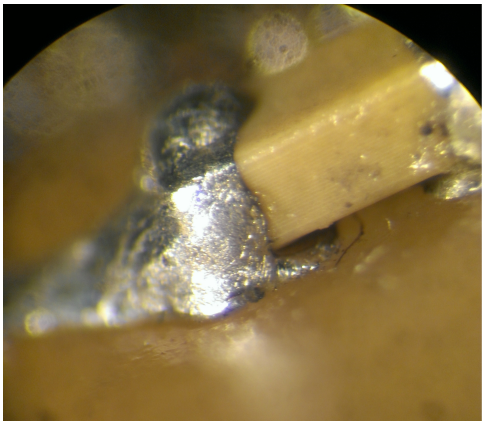


(a) MPU-6050 QFN package soldering. (b) Atmega328P SMD package soldering.

Figure 34: This figure shown how one of the most critical components were soldered on the photolithography-fabricated PCB successfully and accurately



(a) Two 0603 22pF SMD capacitors.



(b) One 0603 1uF capacitor.

Figure 35: This figure shows how passive SMD components were soldered on the PCB board. SMD capacitors with standard size 0603 and resistors with size 0402 were soldered successfully.

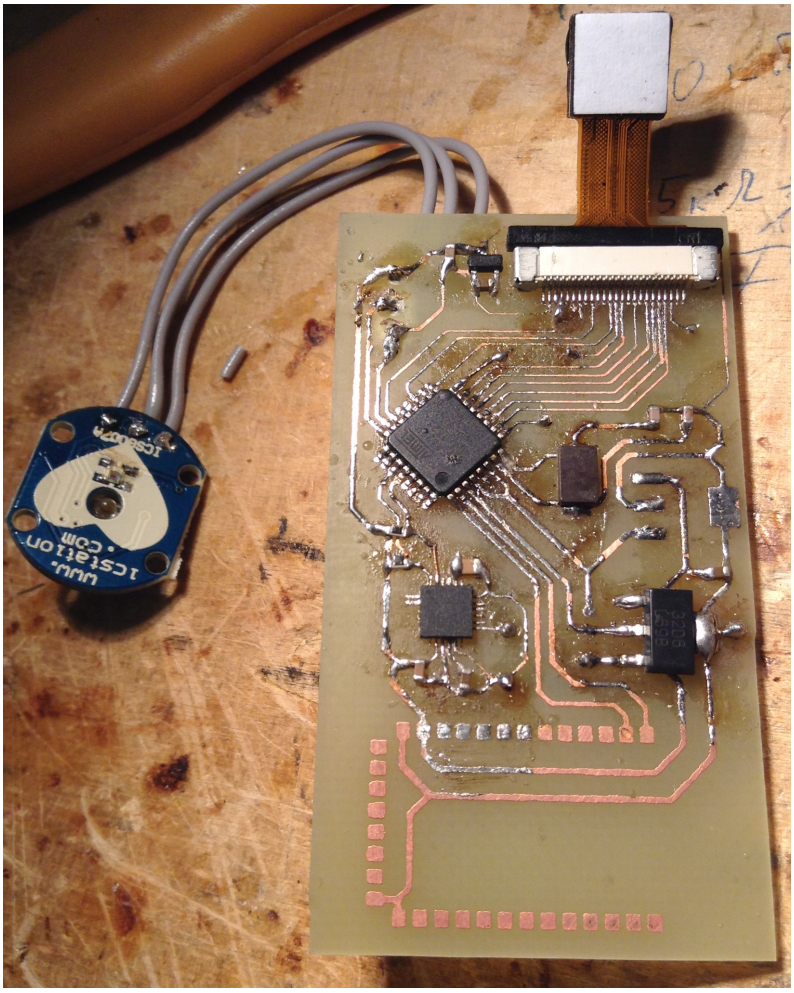


Figure 36: PCB board fabricated by the traditional method of photolithography with most of the components soldered.

4.5 In-chip programming

The possibility of changing the program of the MCU in-chip is fundamental when implementing a prototype. The Atmega328P can be easily programmed by an Arduino board, using the pins 16, 17, 18 and 1, as illustrated in Figure 37. The program is available in Appendix D.

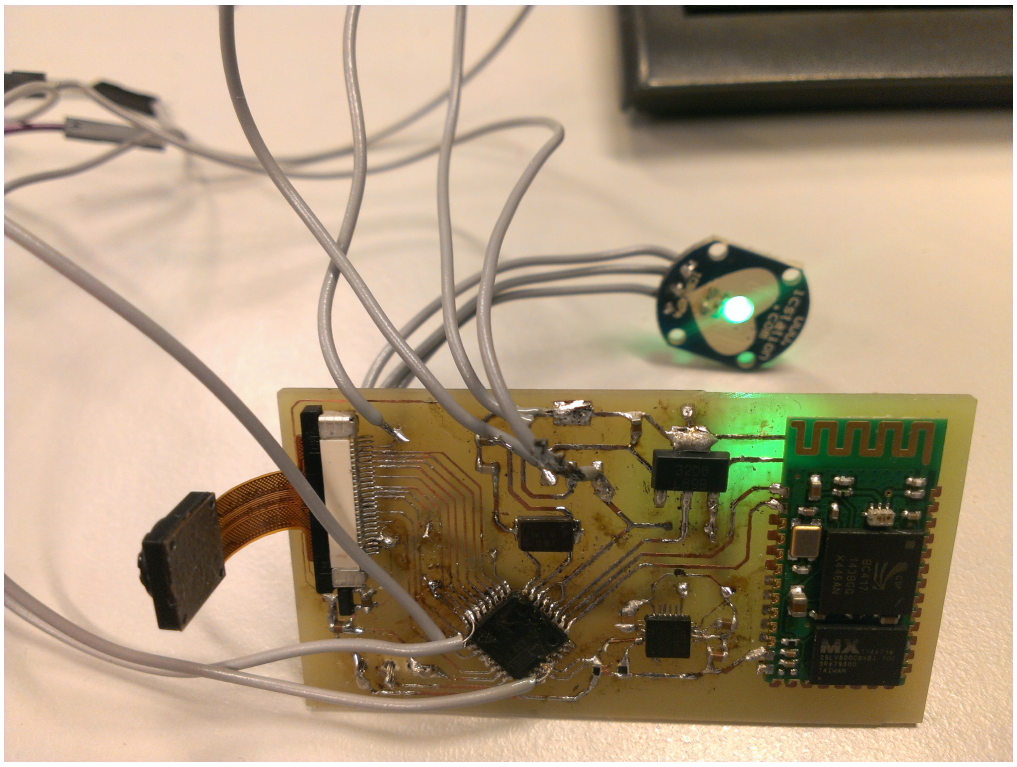


Figure 37: Temporary wires were soldered in pins 1, 16, 17 and 18 of the Atmega328P MCU in order to perform in-chip programming.

4.6 Verifying the design and operation of the device

After the fabrication, soldering of the component and cleaning of the PCB, the operation and the design of the wearable device were verified.

Errors in the circuit connection after fabrication and soldering appear very frequently. For this reason, before powering on the wristband, the board was carefully checked for undesired shortcuts and after that the most critical connections were tested in order to ensure proper operation between the modules.

On power on, the correct values of the voltage outputs of the LDOs were checked for ensuring that every IC is working on the correct voltage. After that, the Bluetooth communication with the Android device has been tested. Once the wireless connection has been done, the sensors data was received in the smartphone correctly, especially the image taken from the digital camera. The device has been put then into debugging mode where the operation of the rest of the components has been verified. First the

integrated camera initialization has been done and after that the information from the MEMS sensors and the optical heart-rate sensor has been received as shown in Figure 38.

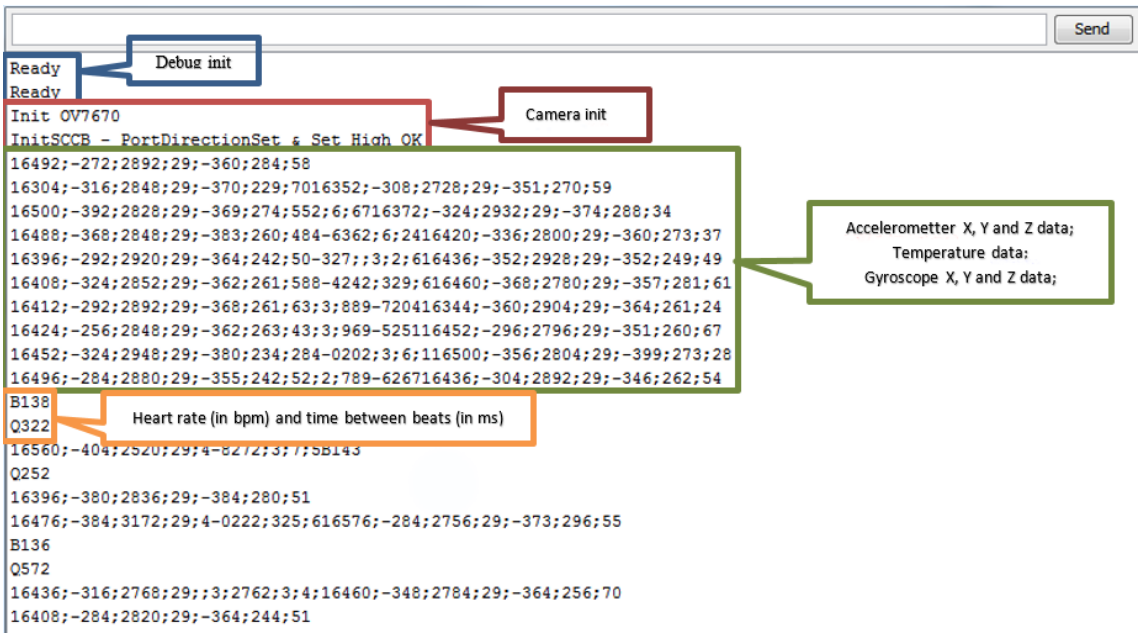


Figure 38: This figure illustrates the verification of the prototype in debug mode. First the camera is initialized and after that information from the sensors is constantly updated.

Finally, the design has been checked as shown in Figure 39. The size of the device resulted to be not small enough to fit perfectly the hand. The main reasons are that the Bluetooth chip is very big and the circuit has been done on non-flexible substrate.

(a) Bottom view. (b) Top view.

Figure 39: Top view and bottom view of the smart wristband prototype.

4.7 Summary

The PRECIOUS wearable design was methodic, following the common design and development techniques.The wearable prototype electrical schematic was designed, choosing cheap and common components. The design was first tested on a prototype board, checking the operation of all the modules. After verification, the PCB layout of the prototype was designed, leaving the board ready for fabrication. On the other hand two PCB fabrication approaches were tested. First the board was fabricated by the method of toner transfer, which is a fast way to fabricated PCB boards and does not require professional environment. However, the resolution of this method was not sufficient for the wearable development because of the 0.2mm line-width of the MPU6050 module pins. For this reason, the wristband circuit was fabricated in the Aalto University Design Factory in a special laboratory, dedicated for the double-layer PCB fabrication by UV-exposure. The results were very good and all the components were soldered.

Moreover, temporary wires were soldered to some of the MCU pins in order to change the program of the device in-circuit. After verifying the PCB fabrication and soldering, the device was tested and the correct operation was achieved after some software bug fixing.

5 Conclusions

Healthcare related smart wristband prototype device has been designed, fabricated and tested. The data sensing and connection with a smartphone has been proved, which allows to measure important human being biometrics and store them in the cloud for future applying of machine learning algorithms or just storing health related information. Thus, the data collected from the wristband and sent to the cloud by a smartphone can be used to create a VIM by contributing to a wide range of physical-related parameters. Also, the information collected by the wearable is processed and presented to the user with the help of the developed Android application.

Furthermore, different fabrication methods have been tested and discussed. On the other hand, the food intake detection and recognition algorithm, that will run on the smartphone after taking a photo of the food with the wristband and sending it via Bluetooth, has been explained and its operation has been demonstrated.

The electrical design of the wearable was straightforward. Only cheap and very common components were used and the reason is that a prototype should be an easy and fast to develop product that proves a specific concept. The goal was not to design a device to be sold on the market, but to demonstrate the advantages of smart wristbands for health-related monitoring.

The price of the materials for building the device was around 20 euro and thus anyone hobbyist can fabricate it. The only issue is linked to the fact that the PCB fabrication under a home environment could not offer very good results and for that reason profession laboratory was needed.

Moreover, there were some issues found that need to be resolved in the future. First, the size of the device needs to be reduced by changing the Bluetooth chip with a smaller one and with better power consumption. Once the size is reduced, fabricating the circuit on a flexible substrate will make the device more comfortable and stylish. Also, the device needs a battery charging circuit for the rechargeable battery, which were not included in the prototype design.

On the other hand, the MCU has to work at 16MHz which leads the operating voltage to be not less than 4.5V. It is the only component that is connected to a 5V power supply. The higher frequency is needed to compute the data from the digital camera, transform it into the correct format and then send it via Bluetooth to the smartphone. There are digital cameras in the market that already offer a dedicated DSP that converts the data and sends it directly via UART[6], meaning that the MCU will just control the power of the camera and therefore no processing will be done by it. Thus, the MCU will be able to operate a 3.3V, minimizing the power and area occupied by the 5V LDO.

There were some issues with the Bluetooth module operation too. The HC06 module clock oscillates at 26MHz and the MCU clock at 16MHz. Therefore, despite the fact that the MCU's UART operates at up to 2Mbaud rate and the Bluetooth module up to 1.3Mbaud, their clock are not synchronized at rates higher than

[6]The communication between the MCU and the Bluetooth device in the wristband prototype is done using the UART protocol.

115200baud. The immediate solution is to change the Bluetooth module. However, after testing a more expensive module which was smaller and supports speeds up to 1Mbaud, we could not achieve Bluetooth connection throughput higher than 90Kbps.

Furthermore, the optical heart rate sensor on the wearable is situated on the lower side of the wrist in order to improve the sensing. This theory is wrong, in fact the upper part of the hand is better for optical sensing of the heart rate and thus the location of the sensors has to be changed.

Despite the mistakes, the wearable was fully operation and able to fulfill the main requirements and goals. In the future, the errors can be easily fixed and some of the modules can be substituted by other more professional and accurate. The board would be then ready to be fabricated on FPC, achieving a product suitable for the market.

References

[1] Dong, Y. *Tracking Wrist Motion to Detect and Measure the Eating Intake of Free-Living Humans.* Clemson University, 2012.

[2] Dong, Y. *Detecting Periods of Eating During Free-Living by Tracking Wrist Motion.* IEEE Journal of Biomedical and Health Informatics (Volume 18 , Issue 4), pp. 1253-1260, 2014.

[3] Ao, B.; Fang, G. and Wang, Y. *Healthcare algorithms by wearable inertial sensors: a survey.* China Communications, (Volume 12 , Issue 4)s (Volume 18 , Issue 4), pp. 1-12, 2015.

[4] He, D.; Liu, C. and Trachanis, J. *Design and simulation of an integrated optical CMOS heart rate sensor.* Advanced Semiconductor Devices and Microsystems (ASDAM), 10th International Conference, pp. 1-4, 2014.

[5] *The PRECIOUS Project.*, 2015. Available: http://www.thepreciousproject.eu/.

[6] Shen, X. *Emerging technologies for e-healthcare* IEE Network (Volume 26, Issue 5) pp. 2-3, 2012.

[7] Liu, H. *SmartCare: Energy-efficient long-term physical activity tracking using smartphones* Tsinghua Science and Technology (Volume 20 , Issue 4), pp. 348 - 363, 2015.

[8] Doron, M. *Estimation of physical activity monitored during the day-to-day life by an autonomous wearable device (SVELTE project)* Engineering in Medicine and Biology Society (EMBC) IEE Conference, pp. 4629 - 4632, 2013.

[9] Shaopeng, L. *Design of a wearable multi-sensor system for physical activity assessment* Advanced Intelligent Mechatronics (AIM) IEE Conference, pp. 254 - 259, 2010.

[10] Shroff, G; Smailagic, A and Siewiorek, D.P. *Wearable context-aware food recognition for calorie monitoring*12h IEEE International Symposium on Wearable Computers, pp. 119 - 120, 2015.

[11] Bi, Y; LV, M and Song, C. *AutoDietary: A Wearable Acoustic Sensor System for Food Intake Recognition in Daily Life* Sensors Journal, IEEE (Volume PP, Issue 99), 2015.

[12] Jindong, L.; Johns, E. and Pettitt, C. *An Intelligent Food-Intake Monitoring System Using Wearable Sensors* Wearable and Implantable Body Sensor Networks (BSN), pp. 154-160, 2012.

[13] Hoilett, O. *Placing pedometer on wrist and transmitting data via bluetooth.* Calvary Engineering Family Group, USA, January 2015.

[14] Scarlett, J. *Enhancing the Performance of Pedometers Using a Single Accelerometer.* Analog Devices, 2007.

[15] Brunvand, E. *Computer Design Lab - VGA driver.*, 2010. Available: http://www.eng.utah.edu/~cs3710/labs/VGA.pdf.

[16] OmniVision Tehcnologies, Inc. *OV7670/OV7171 CMOS VGA Camera Chip Datasheet.*, 2015. Available: http://www.eng.utah.edu/~cs3710/labs/VGA.pdf.

[17] Kong, F. and Jindong, T. *DietCam: Regular Shape Food Recognition with a Camera Phone.* International Conference on Body Sensor Networks, pp. 127-132, 2011.

[18] Kawano, I. and Yanai, K. *Real-Time Mobile Food Recognition System.* Computer Vision and Pattern Recognition Workshops IEEE Conference, pp. 1-7, 2013.

[19] Open Source Computer Vision Library *OpenCV Documentation.*, 2015. Available: http://opencv.org/documentation.html.

[20] Rublee, E.; Rabaud, V. and Konolige, K. *ORB: An efficient alternative to SIFT or SURF.* Computer Vision (ICCV) IEEE International Conference, pp. 2564-2571, 2011.

[21] Puri, M.; Zhiwei, Z. and Qian, Y. *Recognition and volume estimation of food intake using a mobile device.* Applications of Computer Vision (WACV) Workshop, pp. 1-8, 2009.

[22] Farooq, M.; Fontana, J.M. and Boateng, A.F. *A Comparative Study of Food Intake Detection Using Artificial Neural Network and Support Vector Machine.* Machine Learning and Applications (ICMLA) 12th International Conference (Volume 1), pp. 4-7, 2013.

[23] Gitman, Y. *Anatonomy of the DIY heart rate monitor.*, 2012. Available: http://opencv.org/documentation.html.

[24] Alhawari, M.; Albelooshi, N. and Perrott, M.H. *A 0.5V <4μW CMOS photoplethysmographic heart-rate sensor IC based on a non-uniform quantizer.* Solid-State Circuits Conference Digest of Technical Papers (ISSCC) IEEE International Conference, pp. 384-385, 2013.

[25] Loher, T.; Seckel, M. and Pahl, B. *Highly integrated flexible electronic Circuits and Modules.* Microsystems, Packaging, Assembly and Circuits Technology Conference, pp. 86-89, 2008.

[26] Jinsoo, Noh.; Donsun, Y. and Chaemin, L. *Scalability of Roll-to-Roll Gravure-Printed Electrodes on Plastic Foils.* Electronics Packaging Manufacturing, IEEE Transactions (Volume 33, Issue 4), pp. 275-283, 2010.

[27] Kawahara, Y.; Hodges, S. and Nan-Wei, G. *Building Functional Prototypes Using Conductive Inkjet Printing.* Pervasive Computing, IEEE Internation Conference (Volume 13, Issue: 3), pp. 30-38, 2014.

[28] OpenCV. *Face Recognition with OpenCV.*, 2015. Available: http://docs.opencv.org/modules/contrib/doc/facerec/facerec_tutorial.html.

[29] Jun-yan, H.; Yi-lin, C. and Jing, W. *Robust automatic white balance algorithm using gray color points in images.* Consumer Electronics, IEEE Transactions (Volume 52, Issue 2), pp. 541-546, 2006.

[30] Su, J. *Color Balancing Algorithms.*, 2010. Available: http://web.stanford.edu/~sujason/ColorBalancing/.

[31] Feng, L.; Xiaoyu, L. and Yi, C.. *An efficient detection method for rare colored capsule based on RGB and HSV color space.* Granular Computing (GrC), IEEE International Conference, pp. 175-178, 2014.

[32] Arduino. *The Arduino Playground.*, 2015. Available: http://playground.arduino.cc/.

[33] Atmel Corporation *ATmega328/P Datasheet* Rev.: Atmel-ATmega328P-Datasheet_10/2014, 2014.

[34] InvenSense Inc. *MPU-6000 and MPU-6050 Product Specification* Revision 3.2, 2011.

[35] InvenSense Inc. *MPU-6000 and MPU-6050 Register Map and Descriptions* Revision 4.0, 2011.

A Arduino code for OV7670 camera module frame reception

This is the code that is used to configure the camera module, receive the image data from the parallel input data port and send it via UART to the computer or Bluetooth module. Please note that the camera register and configuration files are not available in this document, but only in the online repository https://github.com/TodorGinchev/Arduino/tree/master/OV7670_Todor_Jesus.

```
void loop() {
  pinMode(12, OUTPUT);
  uint16_t i=0, j=0;

  //Wait 3s for the cam to properly init
  _delay_ms(3000);
  while(1){
    //Wait for VSYNC transitiong as beginning of frame
        //Wait for HREF LOW and VSYNC LOW
    while( ((PIND&B00001000)==B00001000) |
                ((PINB&B00000100)==B00000100) );
    //Send start of frame
    for (int cont=0; cont<5; cont++){
      UDR0 = 0x3C; //Send '<' chart via UART
      //PCLK clock delay for sending UART data
      while((PIND&B00000100)==B00000100);//wait PCLK low
      while((PIND&B00000100)!=B00000100);//wait PCLK high
      //PCLK clock delay for sending UART data
      while((PIND&B00000100)==B00000100);//wait PCLK low
      while((PIND&B00000100)!=B00000100);//wait PCLK high
      //PCLK clock delay for sending UART data
      while((PIND&B00000100)==B00000100);//wait PCLK low
      while((PIND&B00000100)!=B00000100);//wait PCLK high
    }
    UDR0 = 0x0A; //Send new line chart via UART
    //PCLK clock delay for sending UART data
    while((PIND&B00000100)==B00000100);//wait PCLK low
    while((PIND&B00000100)!=B00000100);//wait PCLK high
    //PCLK clock delay for sending UART data
    while((PIND&B00000100)==B00000100);//wait PCLK low
    while((PIND&B00000100)!=B00000100);//wait PCLK high
    //PCLK clock delay for sending UART data
    while((PIND&B00000100)==B00000100);//wait PCLK low
    while((PIND&B00000100)!=B00000100);//wait PCLK high

    //Wait till beginning of line
```

```
    //Wait for HREF HIGH and VSYNC HIGH
while( ((PIND&B00001000)!=B00001000) | (
    (PINB&B00000100)!=B00000100) );

// Vertical lines count
for(j=0; j<479; j++){
  //Horizontal pixel position count
    while((PIND&B00000100)==B00000100);//wait PCLK low
    while((PIND&B00000100)!=B00000100);//wait PCLK high
  for(i=0; i<640; i++){
    //PCLK clock delay
    //Read and write data during rising edge
    UDR0 = ((PINC&0x0F)|(PIND&0xF0));
    //UDR0 = i&0xFF;
    //UDR0 = (i&B00011111) | B00100000;//0x00;//0xFF ||
                    //((PINC&0x0F)|(PIND&0xF0));
    //PCLK clock delay
    while((PIND&B00000100)==B00000100);//wait PCLK low
    while((PIND&B00000100)!=B00000100);//wait PCLK high
    //PCLK clock delay
    while((PIND&B00000100)==B00000100);//wait PCLK low
    while((PIND&B00000100)!=B00000100);//wait PCLK high

  }
  //PCLK clock delay for sync next operation of HREF
  while((PIND&B00000100)==B00000100);//wait PCLK low
  while((PIND&B00000100)!=B00000100);//wait PCLK high

  //Wait till HREF down and reevaluate line start
  while( ((PINB&B00000100)==B00000100));//wait HREF low
  PORTB |= B010000; //Turn pin 12 ON
  PORTB &= B101111; //Turn pin 12 OFF

  //Wait till beginning of next line
      //Wait for HREF HIGH and VSYNC HIGH
  while( ((PIND&B00001000)!=B00001000) |
            ((PINB&B00000100)!=B00000100) );
}
//Send end of frame
for (int cont=0; cont<5; cont++){
  UDR0 = 0x3E; //Send '>' chart via UART
  //PCLK clock delay for sending UART data
  while((PIND&B00000100)==B00000100);//wait PCLK low
  while((PIND&B00000100)!=B00000100);//wait PCLK high
  //PCLK clock delay for sending UART data
```

```
            while((PIND&B00000100)==B00000100);//wait PCLK low
            while((PIND&B00000100)!=B00000100);//wait PCLK high
        }
    }
}
```

B Python code that receives the image from the Arduino

This code receives the image data from the Arduino via COM serial communication and stores the information into a file in the computer.

```
##Receive Serial
##Autor: Jes\'us Llorente Santos
##Company: Aalto University, COMNET department
##Date: 19 AUG 2015
##This program is used to receive and store data from a
## serial port
##Usage:
##Open serial communication with COM5 at 2Mbaud speed
## com = connect_serial("COM5", 2000000)
##Read data from the port, store it in a file with name
## log.out
##Stop when timeout=30s is reached. Let the buffer size
##be 4096bytes and flush buffer every 4096*16bytes
##or 16 frames.
## read4(com, "log.out", 30, 4096*16, 4096)
##Close com port.
## com.close()

## import the serial library
import serial

## import time
import time

## Boolean variable that will represent
## whether or not the arduino is connected
connected = False

## Establish connection to the serial port that your
##arduino is connected to.

def connect_serial(dev, baudrate=9600):
    try:
        print "Opening serial port
        % s at %d bps" % (dev, baudrate)
        s = serial.Serial(dev, baudrate)
        return s
    except Exception,ex:
        print "Failed to open serial port
```

```
        % s at %d bps \n%s" % (dev, baudrate, str(ex))
        s.close()

def open_file(filename, mode):
    fd = open(filename, mode)
    return fd

def read1(r_fd, w_fd=None, timeout=0, nflush=1000):
    n = 0
    i = 0
    if timeout != 0:
        ts_stop = time.time() + timeout
        loop    = False
    else:
        ts_stop = 0
        loop = True
    try:
        while((time.time() < ts_stop) | loop):
            data = r_fd.read()
            s = len(data)
            n += s
            i += s
            if w_fd:
                w_fd.write(data)
            else:
                print "[%d] %s" % (n, data)
            if i > nflush:
                w_fd.flush()
                i -= nflush
    finally:
        w_fd.flush()
        print "Read %d bytes" % (n)
        return n

def read2(r_fd, w_fd=None, timeout=0, nflush=1000):
    n = 0
    i = 0
    if w_fd is not None:
        w_fd = open(w_fd, "w")
    if timeout != 0:
        ts_stop = time.time() + timeout
        loop    = False
    else:
        ts_stop = 0
        loop = True
```

```
    try:
        while((time.time() < ts_stop) | loop):
            data = r_fd.read()
            s = len(data)
            n += s
            i += s
            if w_fd:
                w_fd.write(data)
            else:
                print "[%d] %s" % (n, data)
            if i > nflush:
                w_fd.flush()
                i -= nflush
    finally:
        if w_fd is not None:
            w_fd.flush()
            w_fd.close()
        print "Read %d bytes" % (n)
        return n

def read3(r_fd, w_fd=None, timeout=0, nflush=1000):
    n = 0
    i = 0
    if w_fd is not None:
        w_fd = open(w_fd, "w")
    if timeout != 0:
        ts_stop = time.time() + timeout
        loop    = False
    else:
        ts_stop = 0
        loop = True
    try:
        r_fd.flushInput()
        while((time.time() < ts_stop) | loop):
            data = r_fd.read(1024)
            s = len(data)
            n += s
            i += s
            if w_fd:
                w_fd.write(data)
            else:
                print "[%d] %s" % (n, data)
            if i > nflush:
                w_fd.flush()
                i -= nflush
```

```
    finally:
        if w_fd is not None:
            w_fd.flush()
            w_fd.close()
        print "Read %d bytes" % (n)
        return n

def read4(r_fd, w_fd=None, timeout=0, nflush=1000,
nread=1024):
    n = 0
    i = 0
    if w_fd is not None:
        w_fd = open(w_fd, "w")
    if timeout != 0:
        ts_stop = time.time() + timeout
        loop    = False
    else:
        ts_stop = 0
        loop = True
    try:
        r_fd.flushInput()
        while((time.time() < ts_stop) | loop):
            data = r_fd.read(nread)
            s = len(data)
            n += s
            i += s
            if w_fd:
                w_fd.write(data)
            else:
                print "[%d] %s" % (n, data)
            if i > nflush:
                w_fd.flush()
                i -= nflush
    finally:
        if w_fd is not None:
            w_fd.flush()
            w_fd.close()
        print "Read %d bytes" % (n)
        return n

def read_n_write(fd_serial, fd_raw=None, fd_pic=None,
 nflush=1000, nread=1024):
    n = 0
    i = 0
    begin_pic = False
```

```
if fd_raw is not None:
    fd_raw = open(fd_raw, "w")
if fd_pic is not None:
    fd_pic = open(fd_pic, "w")
#Run for your life
ts_stop = 0
loop = True
try:
    fd_serial.flushInput()
    while((time.time() < ts_stop) | loop):
     data = fd_serial.read(nread)
     s = len(data)
     n += s
     i += s
     if "\r\n" in data:
       print "0x0D 0x0A detected"
     if fd_raw:
       fd_raw.write(data)
     if fd_pic:
       #Beginning of frame
       if not begin_pic and ">>>>>><\<\<\<\<\n in data:
         print "Beginning of frame! ", time.time()
         begin_pic = True
         chunk = data.split(">>>>>><\<\<\<\<\n")[1]
         fd_pic.write("<\<\<\<\<\n"+chunk)
       #End of frame
       elif begin_pic and ">>>>>><\<\<\<\<\n" in data:
         print "End of frame! ", time.time()
         chunk = data.split(">>>>>><\<\<\<\<\n")[0]
         fd_pic.write(chunk+">>>>>")
         break
       #Picture data
       elif begin_pic:
         fd_pic.write(chunk)
     else:
       print "[%d] %s" % (n, data)
     if i > nflush:
       if fd_raw:  fd_raw.flush()
       if fd_pic:  fd_pic.flush()
       i -= nflush
finally:
    if fd_raw is not None:
        fd_raw.flush()
        fd_raw.close()
    if fd_pic is not None:
```

```
            fd_pic.flush()
            fd_pic.close()
        print "Read %d bytes" % (n)
        return n

#com = connect_serial("COM5", 2000000)
#read4(com, "log.out", 30, 4096*16, 4096)
#com.close()

#com.close()
#com = connect_serial("COM4", 1000000)
#read_n_write(com, "image.raw", "image.dat", 4096*16, 4096)
```

C Matlab code for processing the input image raw data

This code is used in the computer for converting the image data from different formats such as RGB656 or YUV422 to png format.

```
function [image]=create_image(h,w,filename,format)

%CREATE_IMAGE A function that receives a file with raw image
%   data in different formats and the size of the image. It
%   returns an image matrix and plots the result.
%
%   Autor: Todor Aleksandrov Ginchev Company: Aalto
%   University, COMNET department
%       Date: 25 AUG 2015

%Open File in read mode
fileID = fopen(filename,'r');
image_data=fread(fileID);

%Reserve space for the image matrix zeros(:,:,1) is Red,
%zeros(:,:,2) is Green and zeros(:,:,3) is Blue
image=zeros(h,w,3);

switch(format)
 case 'RGB565'
        %Get RGB565 pixel raw data and store it in a proper
        %way. Remember that the loops star from 0, but
        %array/matrix positions start from 1.
        position_counter=0;
        for i=0:(h-1)
         for j=0:(w-1)
            if(mod(position_counter,479)==0)
                position_counter = position_counter+1;
            end
            %get second byte and shift 8 positions (In
            %OV7670 datasheet this is considered as the
            %first byte...)
            pixel_data = bitshift(image_data(...
                position_counter+1),8);
            %add first byte (In OV7670 datasheet this is
            %considered as the second byte...)
            pixel_data = pixel_data + image_data(...
                position_counter);
```

```
            %RGB565 stores each pixel data in 2bytes.
            %Getting the information can be done by the
            %following way: Red = pixel_data & 0xF800, Green
            %= pixel_data & 0xFE0, Blue = pixel_data &
            %0x001F
            %The following is equal to pixel_data 0xF800 and
            %then shift >>11:
            image(i+1,j+1,1)=bitshift(pixel_data,-11);
                %equal to pixel_data & 0x07E0
            image(i+1,j+1,2)=bitand(pixel_data,2016);
            %equal to shift >>5
            image(i+1,j+1,2)=bitshift(image(i+1,j+1,2),-5);
            %equal to pixel_data & 0x001F
            image(i+1,j+1,3)=bitand(pixel_data,31);
            position_counter = position_counter+2;
            %dec2bin(pixel_data) a=dec2bin(image(i+1,j+1,1))
            %b=dec2bin(image(i+1,j+1,2))
            %c=dec2bin(image(i+1,j+1,3))
        end
    end
case 'RGB555'
    %Get RGB555 pixel raw data and store it in a proper
    %way remember that the loops star from 0, but
    %array/matrix positions start from 1
    position_counter=0;
    for i=0:(h-1)
        for j=0:(w-1)
            if(mod(position_counter,479)==0)
                position_counter = position_counter+1;
            end
            %get second byte and shift 8 positions (In
            %OV7670 datasheet this is considered as the
            %first byte...)
            pixel_data = bitshift(image_data(...
                position_counter+1),8);
            %add first byte (In OV7670 datasheet this is
            %considered as the second byte...)
            pixel_data = pixel_data + image_data(...
                position_counter);
            %RGB565 stores each pixel data in 2bytes.
            %Getting the information can be done by the
            %following way: Red = pixel_data & 0xF800,
            %Green = pixel_data & 0x07E0, Blue =
            %pixel_data & 0x001F
            %equal to pixel_data & 0xF800
```

```
            image(i+1,j+1,1)=bitand(pixel_data,31744);
            %equal to pixel_data & 0x07E0
            image(i+1,j+1,2)=bitand(pixel_data,992);
            %equal to pixel_data & 0x001F
            image(i+1,j+1,3)=bitand(pixel_data,31);
            position_counter = position_counter+2;
        end
    end
case 'RGB444'
    %Get RGB555 pixel raw data and store it in a proper
    %way remember that the loops star from 0, but
    %array/matrix positions start from 1
    position_counter=0;
    for i=0:(h-1)
        for j=0:(w-1)
            if(mod(position_counter,479)==0)
                position_counter = position_counter+1;
            end
            %get second byte and shift 8 positions (In
            %OV7670 datasheet this is considered as the
            %first byte...)
            pixel_data = bitshift(image_data(...
                position_counter+1),8);
            %add first byte (In OV7670 datasheet this is
            %considered as the second byte...)
            pixel_data = pixel_data + image_data(...
                position_counter);
            %RGB565 stores each pixel data in 2bytes.
            %Getting the information can be done by the
            %following way: Red = pixel_data & 0xF800,
            %Green = pixel_data & 0x07E0, Blue =
            %pixel_data & 0x001F
            %equal to pixel_data & 0xF800
            image(i+1,j+1,1)=bitand(pixel_data,3840);
            %equal to pixel_data & 0x07E0
            image(i+1,j+1,2)=bitand(pixel_data,240);
            %equal to pixel_data & 0x001F
            image(i+1,j+1,3)=bitand(pixel_data,15);
            position_counter = position_counter+2;
        end
    end
case 'YUV422'
    %Get RGB555 pixel raw data and store it in a proper
    %way remember that the loops star from 0, but
    %array/matrix positions start from 1
```

```
        for i=0:(h-1)
            for j=0:(w-1)
                pixel_data = image_data(i*w+j+1);
                image(i+1,j+1,1)= pixel_data;
                image(i+1,j+1,2)= pixel_data;
                image(i+1,j+1,3)= pixel_data;
            end
        end
end

fclose(fileID);
figure;
image=uint8(image);
imshow(image);
```

D Wristband prototype source code

```
#include <SoftwareSerial.h>
#include<Wire.h>

//HC-05 BT module declaration
//Connection:
//    Atmega328P  /  HC-05
//            16  /  TXD
//            17  /  RXD
#define RxD 10 //(TxD of the HC-05 module)
#define TxD 11 //(RxD of the HC-05 module)
#define BTenable 5
//MPU-6050 accel/gyro declaration
const int MPU=0x68;//I2C address of the MPU-6050
String AcX,AcY,AcZ,Tmp,GyX,GyY,GyZ;
//OV7670 Camera module declaration
#define SIO_C 2
#define SIO_D 4
#define SIO_CLOCK_DELAY 100
//Bluetooth communication def
SoftwareSerial BTSerial(RxD, TxD);
String inputString = "";
bool stringComplete = false;
//Pulse sensor def//
//Pulse Sensor purple wire connected to A0
int pulsePin = 0;
// pin to blink led at each beat
int blinkPin = 13;
//Volatile Variables, used in the interrupt service routine
volatile int BPM;// holds raw Analog in 0, update every 2mS
volatile int Signal;// holds the incoming raw data
//holds the time interval between beat
volatile int IBI = 600;
// "True" when User's live heartbeat is detected.
//"False" when not a "live beat".
volatile boolean Pulse = false;
//Becomes true when Arduino finds a beat.
volatile boolean QS = false;
//Regards Serial OutPut -- Set This Up to your needs
//Set to 'false' by Default. Re-set to 'true'
//to see Arduino Serial Monitor ASCII Visual Pulse
static boolean serialVisual = false;
```

```
void setup()
{
  //MPU-6050 Init
  Wire.begin();
  Wire.beginTransmission(MPU);
  Wire.write(0x6B);//PWR_MGMT_1 register
  Wire.write(0);//set to zero (wakes up the MPU-6050)
  Wire.endTransmission(true);
  //BT init
  pinMode(BTenable, OUTPUT);
  // Turn on the HC-05 Bluetooth module
  digitalWrite(BTenable, HIGH);
  delay(300);
  //Start serial communication with BT module
  BTSerial.begin(9600);
  BTSerial.flush();
  delay(500);
  // Start serial communicatino with PC (debug)
  Serial.begin(9600);
  Serial.println("Ready");
  //OV7670 init
  Serial.println("Init OV7670");
  if(InitOV7670())
    Serial.println("InitOV7670 OK");
  else
    Serial.println("InitOV7670 NG");
  delay(500);
  //Pulse sensor init
  // pin that will blink to your heartbeat!
  pinMode(blinkPin,OUTPUT);
  // sets up to read Pulse Sensor signal every 2mS
  interruptSetup();
}

void loop()
{
  //Check for BT data.
  while (BTSerial.available())
    readBTinput();
  //Read and store sensors information
  storeSensorData();
  getPulse();
}
```

```
//
// Read sensor data and store it
//
void storeSensorData(){
   Wire.beginTransmission(MPU);
   // starting with register 0x3B (ACCEL_XOUT_H)
   Wire.write(0x3B);
   Wire.endTransmission(false);
   // request a total of 14 registers
   Wire.requestFrom(MPU,14,true);
   AcX=(String)(Wire.read()<<8|Wire.read());
   AcY=(String)(Wire.read()<<8|Wire.read());
   AcZ=(String)(Wire.read()<<8|Wire.read());
   int16_t aux=(Wire.read()<<8|Wire.read());
   aux = aux/340.00+36.53;
   Tmp = (String)aux;
   GyX=(String)(Wire.read()<<8|Wire.read());
   GyY=(String)(Wire.read()<<8|Wire.read());
   GyZ=(String)(Wire.read()<<8|Wire.read());
   String data = AcX+";"+AcY+";"+AcZ+";"+Tmp+
       ";"+GyX+";"+GyY+";"+GyZ;
   Serial.println(data);
   BTSerial.println(data);
}
//
// Read Bluetooth data
//
void readBTinput(){
  char inChar = BTSerial.read();
  Serial.println(inChar);
  // BTSerial.flush();
   //Store until '\n' is received
   //but be careful with buffer overflow
   if (inChar == '\n') {
     stringComplete = true;
   }
   else {
     if(inputString.length()>20)
       inputString="";
     inputString += inChar;
   }
   if (stringComplete && inputString.startsWith
    (">RELAY;") && inputString.endsWith("<;\r")){
     inputString = inputString.substring(inputString.
        indexOf(';')+1,inputString.length());
```

```
      Serial.println(inputString);
      int relayNum = inputString.substring(0,inputString.
         indexOf(';')).toInt();
      inputString = inputString.substring(inputString.
       indexOf(';')+1,inputString.length());
      bool relayState=(inputString.substring(0,inputString.
        indexOf(';')).equals("ON"))? true : (inputString.
          substring(0,inputString.indexOf(';')).equals
          ("OFF"))? false : relayState; //Safe state is OFF
    }
   else if (stringComplete && inputString.startsWith
      (">HUMCTRL;") && inputString.endsWith("<;\r")){
     inputString = inputString.substring(inputString.
        indexOf(';')+1,inputString.length());
      Serial.println(inputString);
      int enableHumidity = inputString.substring(0,
         inputString.indexOf(';')).toInt();
    }
   //Clear data
  inputString = "";
  stringComplete=false;
}

void getPulse(){
  serialOutput() ;
  if (QS == true){
  //A Heartbeat Was Found
  //BPM and IBI have been Determined
  //Quantified Self "QS" true when arduino finds a heartbeat
    digitalWrite(blinkPin,HIGH);
    serialOutputWhenBeatHappens();
    QS = false;
  }
  else {
        digitalWrite(blinkPin,LOW);
  }
  delay(20);//  take a break
}
```